175 PTAS
68
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
SALTER
COMPAQ
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VI-Fascículo 68

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-034-7 (tomo 6)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 018505

Impreso en España-Printed in Spain-Abril 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Lo que vendrá

Trabajador de cuello metálico

Sensores
Infrarrojos, luminosos, etc.

Antena
Comunicaciones a distancia

Unidad de disco
Para entrada/salida de datos y software

Mano herramienta de potencia

Pantalla y teclado
El robot es un ordenador móvil y se lo puede utilizar como tal; los ingenieros de mantenimiento necesitarán acceder al software y programas de diagnóstico del robot

Salida impresa
Produce recibos, horarios, instrucciones

Espacio de carga/almacenamiento

Altavoz
Ofrece una salida de habla sintetizada

Sondas
Gama de instrumentos de medición, lápices ópticos, sondas lógicas

Micrófono
Para reconocimiento de voz y sonido

Pie
El robot se puede "sentar" sobre su base, puede hacer girar sus piernas y cambiar sus pies por pinzas, cambiando su capacidad de movimiento por una enorme manejabilidad y fuerza

Steve Cross

Operario del mañana
Si alguna vez se desarrollara un robot con fines generales parecido al ser humano, a un costo que lo convirtiera en un sustituto razonable de la mano de obra semiespecializada, necesitaría una "inteligencia" sumamente desarrollada que comprendiera base de datos de conocimientos, integración sensorial, base de datos de habilidades y software para aprendizaje. Una inteligencia como ésta se podría empaquetar en una gran diversidad de cuerpos. He aquí uno de los posibles tipos, que podría funcionar como un obrero semiespecializado de la industria ligera o pesada

Finalizaremos esta serie dedicada a la robótica tratando de vislumbrar los futuros adelantos en este campo

Nuestra serie sobre robótica ha dejado claro cómo el mundo real de los robots continúa muy alejado del concepto que ha creado la literatura novelesca acerca de los seres pensantes mecánicos. Nuestra imaginación nos ha conducido a esperar ciertas cosas de los robots. Esperamos que sean capaces de moverse con entera libertad, alimentados con su propia energía; que vean, oigan y perciban el mundo que los rodea; que hablen con nosotros sobre ciencia y filosofía o que, al menos, se comuniquen de una forma inteligente, y que manejen objetos e ideas tal como lo hacemos nosotros. En otras palabras, en nuestra imaginación hemos creado los robots a nuestra propia imagen y semejanza. Cuando observamos con ojo crítico los robots comerciales, industriales y para aficionados existentes en la actualidad, a menudo nos sorprendemos de lo bien que pueden realizar sus tareas específicas, sintiéndonos, al mismo tiempo, algo decepcionados por el hecho de que no puedan hacer más.

Sabiendo lo que ya sabemos ahora acerca de la naturaleza del diseño y la implementación de ro-

Variaciones visuales

El conocimiento acerca de un objeto se puede almacenar como una imagen "modelo" del objeto arquetípico, más una serie de sentencias de datos de variación; cada sentencia de variación se puede aplicar al modelo para producir una imagen diferente que aún se ajuste a la definición del tipo. Una imagen captada por los sensores del robot se explora mediante un módulo de análisis bruto que proporciona una primera hipótesis respecto a la clase de objeto que está a la vista. Cada objeto se compara con la imagen recibida hasta hallar una pareja. La confianza estadística con la cual se efectúe este emparejamiento determina si las variaciones de la imagen exigen la generación de una nueva sentencia de datos de variación

bots, ¿qué podemos esperar, realista y prácticamente, de los robots del futuro?

Movimiento

Es sumamente improbable que en un futuro cercano los robots caminen apoyados en algo que se parezca a una extremidad humana. Sería preciso destinar demasiado espacio de memoria y tiempo de proceso al esfuerzo que supone mantener el equilibrio, mientras que las junturas y la musculatura eléctrica o hidráulica carecen de la flexibilidad o la libertad de que gozan los seres humanos gracias a la interacción de músculos, tendones y cartílagos. Además, existen muchas ocasiones en las que el robot se vería muy limitado por el hecho de tener que andar sobre dos piernas. No obstante, recientemente se han construido algunos robots experimentales con 4 o 6 extremidades, con apariencia de insectos. Éstos podrían ofrecer una interesante variación de diseño para algunas aplicaciones robot.

Imagen recibida

Actualización base de datos

Base de datos de conocimientos

Clase #512: Recipientes

Módulo de reconoc. bruto

Modelo: Cazuela
Modelo: Vaso
Modelo: Botella
Modelo: Tetera
Modelo: Bol
Modelo: Cacerola
Modelo: Jarra
Modelo: Taza

Determ. de pareja

SALIDA DE PROPORCIÓN DE CONFIANZA

Es PC>51 % ?

N

No hay pareja Proseguir búsqueda

S

Es PC<80 % ?

N

Pareja familiar Fin búsqueda

S

Generador de imagen diferente

Nuevos datos de variación

Ian McKinnell

Para otras aplicaciones, como pueden ser las operaciones militares, la exploración sideral y muchos usos convencionales en el hogar, las ruedas suelen proporcionar el método más práctico de movimiento, y es poco probable que esta situación se modifique. El desplazamiento del robot se volverá más fluido, pero probablemente jamás pueda competir con la belleza y la armonía de un atleta humano en movimiento.

Muchos robots industriales y brazos-robot más pequeños están necesariamente fijos en un sitio, con sus movimientos restringidos a un área de acción muy específica, dado que están diseñados para llevar a cabo una o dos tareas bien definidas. A menos que la naturaleza del montaje industrial cambie de forma radical, es probable que incluso los robots industriales ambulantes continúen relativamente confinados: seguirán siguiendo huellas, desplazándose sobre raíles o colgando de guías elevadas. Es posible que los avances en el campo de la automatización y el diseño robótico den lugar a un cambio radical en los métodos de producción industrial, pero es imposible predecir en qué consistirá.

Un elemento clave del movimiento es la necesidad del robot de responder a su entorno. Esto significa que el robot debe estar bien equipado con un sistema sensorial y que la entrada sensorial debe estar en relación directa con su movimiento.

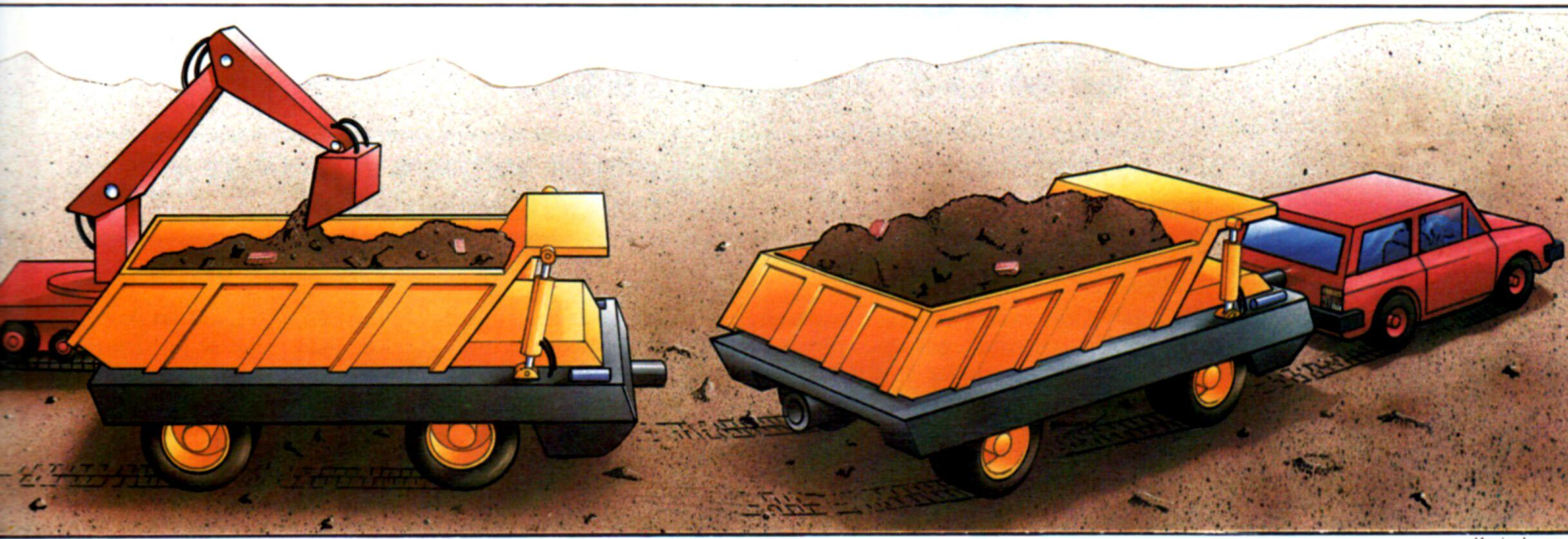
Sensores

Se puede dotar a los robots de equipos sensoriales sofisticados que amplíen sus percepciones a áreas nuevas o desconocidas para los humanos. Los sensores de proximidad, los detectores de movimiento, los dispositivos de realimentación posicional discreta y precisa y los detectores de ruido con una gama muy amplia de frecuencias perceptibles, le confieren a un robot la capacidad de captar datos más variados que los que puede recoger un ser humano. Los sistemas visuales se están volviendo más exactos, con un aumento de la resolución de las imágenes percibidas. Las técnicas de síntesis y reconocimiento de voz serán, ciertamente, cada vez más sofisticadas y desempeñarán un importante papel en el desarrollo de la robótica.

Es muy probable que los robots de utilidades, empleados principalmente para aplicaciones industriales, continúen estando dotados sólo de aquellos sensores necesarios para llevar a cabo las tareas que se les han asignado. Los brazos-robot soldadores, por ejemplo, no tienen ninguna necesidad de disponer de voz ni de una compleja realimentación visual. Pueden llevar a cabo sus trabajos con precisión y rapidez con un mínimo de entrada sensorial, y las percepciones externas posiblemente constituirían más bien un obstáculo.

Los robots para fines generales, diseñados para aprender a partir de la experiencia e imitar los procesos del pensamiento humano, habrán de estar equipados con la mayor cantidad de sensores posible. Sería crucial que el robot fuera capaz de investigar su entorno independientemente y asimilar la información recopilada. Los seres humanos dependen en gran medida, por ejemplo, de una combinación de realimentación visual y auditiva para comprender el habla. Nosotros a menudo tendemos a ignorar esta síntesis de sensaciones, en particular cuando pensamos en términos del diseño de robots.





Kevin Jones

Convoy de anticipación
Es probable que las técnicas de la robótica produzcan su mayor impacto en la sociedad cuando se las incorpore a herramientas de grado inferior para fines especiales tales como grúas, máquinas excavadoras, vehículos de reparto local y transporte pesado. En la ilustración vemos una máquina excavadora robot cargando un tren de camiones robot en una obra en construcción. Cuando está cargado, cada camión avanza semiinteligentemente hasta un punto de ensamblaje y se engancha por sí mismo a un tren de camiones. El tren circula por las carreteras gracias a la energía de los camiones individuales que lo componen, pero controlado por el conductor humano del vehículo guía. La combinación de la capacidad humana de decisión y dirección con la fuerza bruta y la primaria inteligencia de los robots probablemente redundará en una utilización más racional y rentable de los recursos existentes y de la tecnología del futuro

Pero para que un robot se comunique con un ser humano, para entender el lenguaje en vez de tan sólo reconocerlo, esta combinación de vista y sonido resulta esencial.

En la actualidad, la cantidad de datos que un robot puede aceptar y manejar está severamente restringida por la cantidad de memoria necesaria para almacenar la entrada sensorial, y por las limitaciones de la potencia y la velocidad de proceso. Un robot puede almacenar una imagen visual de un objeto, como puede ser una manzana, y relacionar la imagen con un nombre. El almacenamiento de la imagen ocupa memoria y, cuanto mejor sea la resolución visual del robot, más memoria se necesitará para guardar la imagen. Puesto que, además, todas las manzanas no son iguales, el robot debe tener memoria suficiente para almacenar un muestreo característico de imágenes de manzanas, o bien un algoritmo que reconozca las variaciones y sea capaz de hacer girar la imagen básica de modo que la manzana se pueda contemplar desde cualquier posición. Aun con un mínimo nivel de resolución (p. ej., 256 pixels por imagen), la cantidad de variaciones puede muy bien ser de varios millares.

En el futuro es probable que las demandas de memoria se satisfagan mediante chips de RAM de mayor capacidad (actualmente se están desarrollando chips de 1 Mbit) y a través de la utilización de chips de RAM exclusivos que almacenen datos "de variaciones". El procesador puede ir llamando a los datos de fines generales retenidos en estos chips a medida que sea necesario para clarificar diversas imágenes diferentes.

Además de una enorme memoria, la auténtica conciencia sensorial de un robot requeriría que los datos entraran simultáneamente desde muchas fuentes y que pudieran ser procesados rápidamente. Los procesadores existentes serían incapaces de tratar todo el volumen de información que entraría en cualquier momento dado, y enseguida comenzarían a amontonarse datos en espera de ser procesados. Una solución muy probable para este problema es el empleo de dos o más procesadores de gran velocidad y capacidad trabajando en paralelo. Un procesador controlador actuaría entonces como administrador, distribuyendo tareas entre los procesadores del sistema que estuvieran desocupados.

El progreso hacia la resolución de los problemas de hardware con que se encuentran los investigadores se está produciendo a gran velocidad. Pero un robot necesitará de un software muy complejo que le permita comprender lo que se está procesando. Es decir, el robot necesita una mente para saber lo que tiene que hacer con sus percepciones.

La mente

Tal como hemos visto al analizar el movimiento y los sensores, hay dos direcciones principales para la robótica. La primera, y la que cuenta con mayores probabilidades de explotarse pronto, es el área de las herramientas inteligentes, o robots de utilidades. Los brazos industriales y los sistemas de fabricación automatizada que llevan a cabo tareas específicas, independientemente de lo complejas que éstas sean, sólo necesitan estar provistos de un software controlador cuidadosamente definido. A un brazo robot se le puede dar un juego de coordenadas y programarlo para que ejecute una secuencia de acciones sin comprender lo que está haciendo, dónde se encuentra, ni ningún otro detalle relativo a su entorno. El resultado podría ser una puerta perfectamente pintada en una cadena de montaje de coches, o un coche muy bien lavado en una cadena automatizada de lavado de automóviles. Sin embargo, en la medida en que se les solicite a los robots la realización de una mayor diversidad de tareas, éstas estarán necesariamente definidas con menos claridad. Si tienen que desplazarse por una habitación en la cual el contenido cambia día a día, deben ser capaces no sólo de reunir y procesar información, sino también de incorporar nuevas percepciones en su comprensión del mundo. Al robot se le debe proporcionar software operativo y controlador, pero es preciso que haya lugar para que éste se incremente.

El tratar de crear una mente mecánica plantea importantes cuestiones, que de momento aún están sin resolver, acerca de la forma en que piensan y aprenden los humanos. Por ejemplo, ¿qué sabe una criatura en el momento de nacer? El humano ya adulto, ¿es enteramente producto de su entorno o de su herencia, y cuál es la relación entre éstos? ¿Parte el humano con un conjunto de construcciones internas que lo ayudan a aprender lengua, matemáticas, etc.?; de ser así, ¿cómo funcionan? Es difícil responder a estas preguntas sin poder experimentar en el cerebro humano.

Poder mental

Al "BrainStorm", avanzado paquete de software vertical, se le considera el primer "procesador de pensamientos"

El *BrainStorm* ha sido calificado como un "procesador de pensamientos", pero esto no significa que el programa intente emular al cerebro. Lo que hace realmente el *BrainStorm* (literalmente, "idea genial") es ayudar al usuario a organizar sus pensamientos. Será útil una analogía con los métodos preelectrónicos para comprender qué hace.

Cuando se planifica cualquier proyecto complejo, resulta útil confeccionar listas de lo que se tiene que hacer. Una lista de objetivos principales generará sublistas de cómo completar cada punto de la lista precedente, y así sucesivamente hasta que el planificador queda totalmente mareado de tanto andar revolviendo trozos de papel. Cualquier cambio (p. ej., decidir que uno de los puntos de la lista principal en realidad debe incluirse en la sublista de otro punto) puede significar tantos borrones y enmiendas que el sistema se vuelve inmanejable. *BrainStorm* hace que esta clase de cambios y desarrollos sean tan sencillos como las funciones de "recortar y pegar" de que disponen los procesadores de textos; de modo que, al fin y al cabo, la expresión "procesador de pensamientos" tampoco está tan errada.

Cualquier elemento de cada lista o sublista puede ser *promovido*, convirtiéndose en el título de una sublista inferior; y, si es necesario enlazar entre sí actividades similares de listas diferentes, se les puede dar el mismo nombre, aludiendo a ellos como *tocayos*. En ese caso, todo lo que se añada a una sublista encabezada por un tocayo le será añadido también a las otras listas que utilizan el tocayo.

Quienes estén familiarizados con las más bien hostiles facilidades de edición de un paquete para tratamiento de textos como el *WordStar*, encontrarán que es muy fácil acostumbrarse a las instrucciones para control del cursor no mnemotécnicas. Pulsando la tecla Control (CTRL) conjuntamente con la tecla S, el cursor se mueve hacia la izquierda. CTRL–D, CTRL–E y CTRL–X mueven el cursor hacia la derecha, arriba y abajo, respectivamente, y estas cuatro teclas forman en el teclado un patrón lógico. De todos modos, estas teclas se pueden redefinir. A lo que sí resulta menos sencillo habituarse es al hecho de que uno puede desplazar el cursor hacia arriba y hacia abajo mientras está entrando texto, pero tiene que pasar a la modalidad "corrección" (utilizando CTRL–A) para desplazarse hacia la izquierda o la derecha de una línea.

Creación de un esbozo

Generación de sublistas

Operación del programa

El menú de apertura del *BrainStorm* ofrece 11 opciones, cada una de las cuales se selecciona mediante una letra inicial; la mayoría de éstas se explican por sí solas: U*se (usar)*, Load (cargar), Print (imprimir), ID Drive (para pasar a una unidad diferente de la unidad por defecto), Clear (limpiar, para borrar de la memoria el modelo actual), Save (guardar), Write (escribir, para grabar en disco), Directory (directorio), Xit, Merge (mezclar) y Kill.

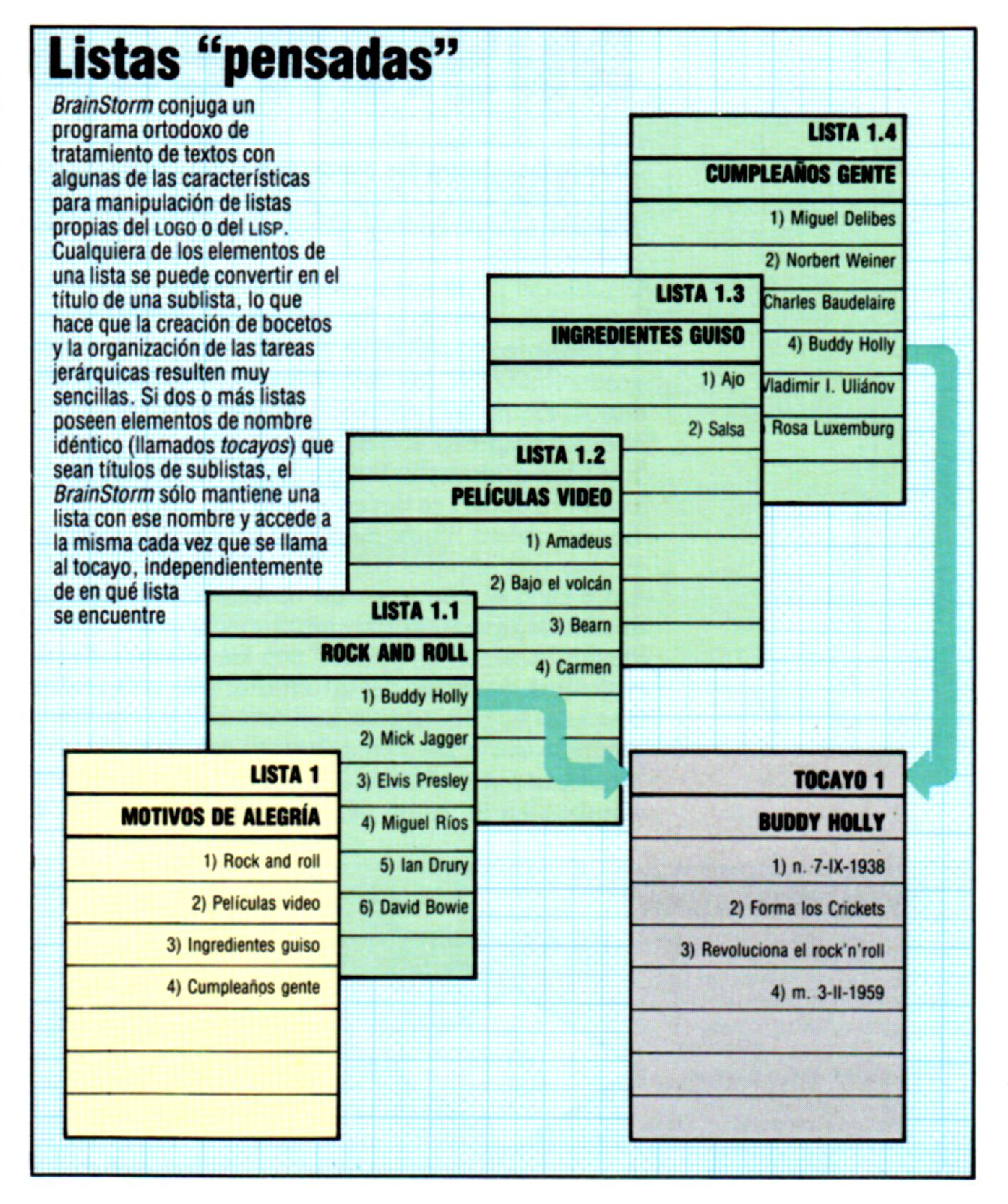
Para empezar, el usuario pulsa U y el programa entra inmediatamente en modalidad de teclado. Se pueden entrar ideas como listas más o menos aleatorias (lo que se conoce como un *model*: modelo). P. ej., un modelo podría estar compuesto por:

```
Leer manual
Comenzar a teclear lista
Teclear sublista
Editar lista
```

Las instrucciones de control no aparecen de forma inmediata, pero se listarán pulsando ?. Desplazando el cursor hacia arriba hasta cualquiera de los elementos de la lista y pulsando CTRL–R, se "promueve" el elemento designado, el cual pasa a ser el encabezamiento de una sublista. Del mismo modo, cualquier elemento de esa sublista se puede convertir en el encabezamiento de una nueva sublista, y así sucesivamente hasta agotar la memoria. Para retornar a la lista anterior, se pulsa CTRL–C.

Los elementos se pueden trasladar, ya sea dentro de una misma lista como a listas de otros niveles, etiquetándolos con el signo @ y utilizando luego CTRL–G (de Get: tomar) o CTRL–P (de Put: poner) para ejecutar el movimiento. Después de emplear CTRL–G, el signo @ pasa automáticamente al siguiente elemento de la lista; por consiguiente, volviendo a pulsar CTRL–G se pasará el elemento siguiente, y así sucesivamente. Ésta es una valiosa facilidad para trasladar una serie completa de elementos a una sublista inferior o superior.

Si se desea insertar nuevos elementos en una lista ya existente, sólo hay que desplazar el cursor hasta el comienzo de la línea que ha de ir tras el elemento nuevo, y entrarlo. Cuando se pulsa RETURN el resto de la lista se desplaza una línea hacia abajo para hacerle sitio. Pulsando CTRL–A (de Amend: corregir) se puede modificar cualquier elemento.



Listas "pensadas"

BrainStorm conjuga un programa ortodoxo de tratamiento de textos con algunas de las características para manipulación de listas propias del LOGO o del LISP. Cualquiera de los elementos de una lista se puede convertir en el título de una sublista, lo que hace que la creación de bocetos y la organización de las tareas jerárquicas resulten muy sencillas. Si dos o más listas poseen elementos de nombre idéntico (llamados *tocayos*) que sean títulos de sublistas, el *BrainStorm* sólo mantiene una lista con ese nombre y accede a la misma cada vez que se llama al tocayo, independientemente de en qué lista se encuentre

Al darle el mismo nombre (p. ej., una fecha) a elementos de listas diferentes, éstos se convierten en *tocayos* y automáticamente se establece entre ellos una referencia cruzada. De modo que si una lista dada exigiera que determinado acontecimiento sucediera en un día especial, supongamos el 1 de enero, se podría crear una lista de acontecimientos para esa fecha y ello permitiría actualizar continuamente un horario de las actividades del día mediante la referencia cruzada.

Se puede crear cualquier cantidad de tocayos y acceder después a ellos en secuencia, utilizando CTRL–S para la ocurrencia siguiente y CTRL–D para la previa. Esto se conoce como una lista doblemente encadenada circular, de modo que al final de la última ocurrencia CTRL–S retornará a la primera.

Las listas, por supuesto, se pueden imprimir y, por lo general, se las formatea con indentaciones que representan los niveles de la lista. Por ejemplo:

- Leer manual
  - Sacar el manual de la caja
    - Buscar índice
      - Hallar página requerida
    - Leer página
  - Volver a poner el manual en la caja
- Empezar a entrar lista
  - Pulsar CTRL – R para promover el elemento de la lista como un nuevo encabezamiento
    - Digitar sublista
  - Pulsar CTRL – C para retornar a la lista anterior
- Editar lista
  - Pulsar CTRL – A para modificar una entrada

El tamaño del indentado lo especifica el usuario.

También se pueden editar listas desde fuera del *BrainStorm*. Si han sido salvadas utilizando la instrucción *Write-to-disk* (grabar en disco), el *WordStar* y otros programas para tratamiento de textos las reconocerán como documentos, y se las podrá editar, imprimir y, de ser necesario, volver a guardar. Esto significa, por ejemplo, que se puede utilizar el *BrainStorm* para preparar la sinopsis de un libro, que puede ser luego escrito utilizando un programa normal para tratamiento de textos, accediendo al archivo BrainStorm.Doc a medida que avanza el libro.

Gracias a la flexibilidad del programa, este proceso puede comenzar a partir de los primeros apuntes e ideas generales (nuevamente, algo que por lo general se realiza en trozos de papel) que después se convertirán en títulos de capítulos (bajo los cuales se listan los contenidos). Si un elemento de un capítulo se vuelve lo suficientemente extenso como para convertirse en un capítulo separado, esto se puede realizar con suma facilidad. Del mismo modo, se pueden "degradar" títulos de capítulos que resulten ser menos importantes de lo que se pensó en un primer momento.

De esta forma, el esbozo general del libro empieza a surgir y, dado que el *WordStar* o programas similares pueden acceder a archivos del *Brain Storm*, la transformación de este esbozo en el manuscrito acabado va fluyendo naturalmente a partir de los primeros procesos intelectuales. Esto significa que no es necesario adoptar los procedimientos, bastante laboriosos, del *BrainStorm*, si uno desea, por ejemplo, imprimir a dos o tres espacios.

El programa viene con un modelo de muestra (llamado SAMPLE.BRN), que incluye el esqueleto de un planificador de horarios, archivo de nombres y direcciones, una lista de tareas (distribuidas en sublistas de "urgentes", "importantes" y "no olvidar"), más una sección para notas. Este modelo es bastante valioso y puede ser añadido a los modelos propios mediante el empleo de la opción *Merge*.

Aprender a usar el *BrainStorm* es fácil. El manual de hojas sueltas es muy claro (fue escrito utilizando el *BrainStorm*), pero las instrucciones siempre están a disposición del usuario en el menú de la pantalla, por lo que la referencia continua al manual resulta innecesaria.

No todas las instrucciones son mnemotécnicas, por lo que algunas podrían resultarle difíciles de recordar al usuario; pero se proporciona un programa INSTALLB, que permite volver a configurar todas las instrucciones y alterar los menús para adaptarlos a la nueva estructuración. Este programa es muy claro y totalmente activado por menú.

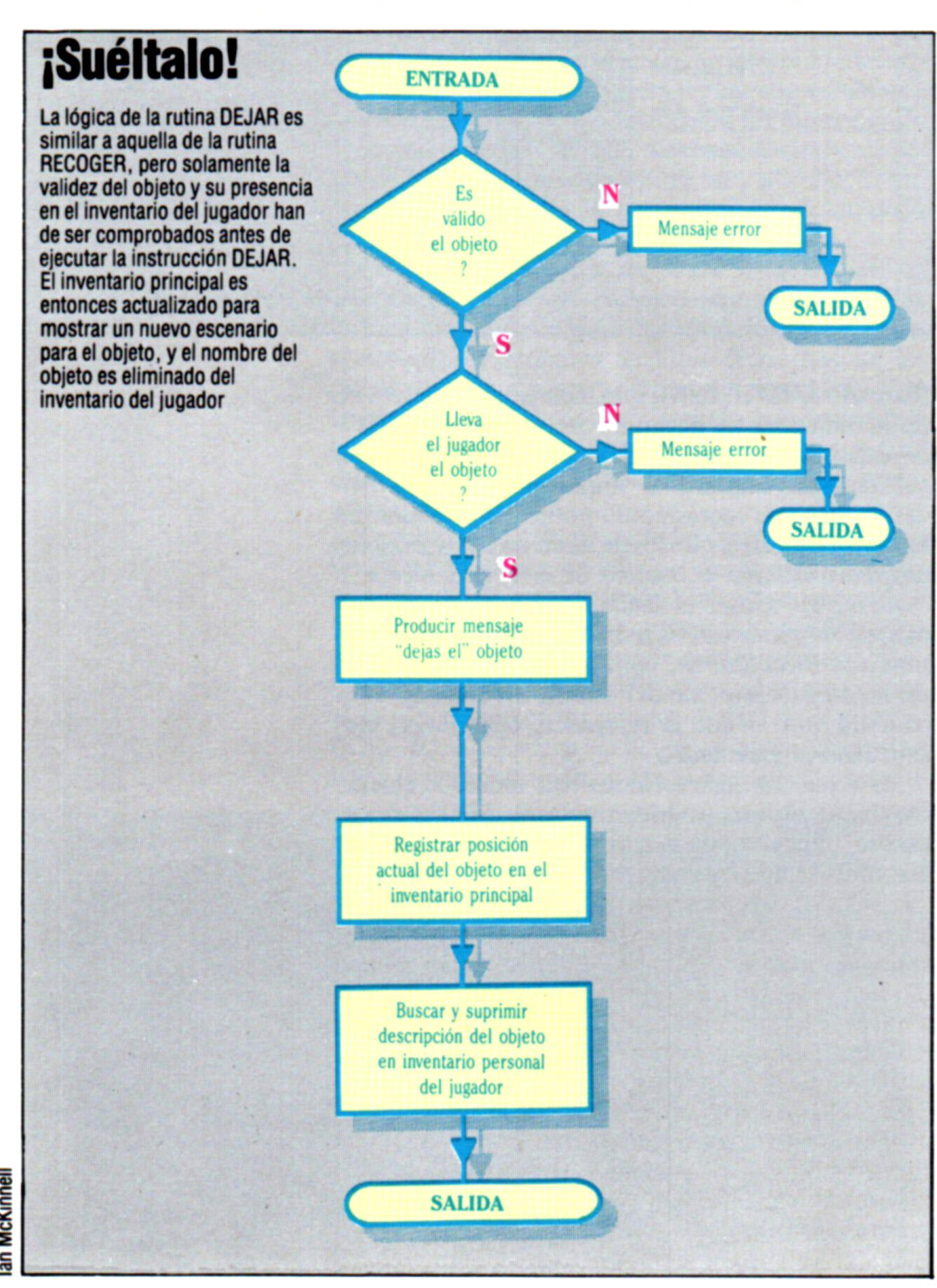
BrainStorm: Para unas 25 máquinas CP/M, MS-DOS y PC-DOS, incluyendo IBM, Sirius y Apricot
Editado por: Caxton Software Ltd., 10-14 Bedford Street, London WC2E 9HE, Gran Bretaña
Autores: David Tebbutt y Mike Liardet
Formato: Disco

Misión especial

Prosiguiendo con el proyecto, desarrollaremos las rutinas para dejar objetos y nos ocuparemos de los escenarios especiales

La subrutina DEJAR tiene muchas similitudes con la rutina RECOGER que describimos en el capítulo anterior. De hecho, podemos utilizar las mismas rutinas de comprobación de objetos que se desarrollaron para utilizar con la instrucción RECOGER. En la rutina RECOGER se llevan a cabo tres comprobaciones del objeto. La primera está diseñada para comprobar si la segunda parte de la sentencia de la instrucción contiene o no un objeto válido. Esto se realiza comparando sistemáticamente cada palabra de la frase de la instrucción con los nombres de los objetos de la matriz del inventario, IV$(,). De hallar una pareja, se establece una variable, F, que da la posición del objeto emparejado dentro de la matriz. Esta comprobación de validez también se debe emplear en la rutina DEJAR para comprobar si el objeto existe y, de ser así, determinar su posición en el inventario.

La segunda comprobación utilizada en la rutina RECOGER también se emplea en la rutina DEJAR; ésta comprueba si el jugador lleva el objeto especificado en la instrucción en el inventario de objetos transportados, IC$(). Obviamente, ¡el jugador no puede dejar un objeto que no esté llevando consigo! La tercera prueba utilizada en la rutina RECOGER efectúa una comprobación para asegurar que el objeto a recoger esté en el escenario en curso del jugador, determinado por P, la variable de posición. Sin embargo, dado que el objeto a dejar debe estar en posesión del jugador, su posición no aparecerá en el inventario principal y, por consiguiente, esta tercera prueba no es necesaria en la rutina DEJAR.



¡Suéltalo!

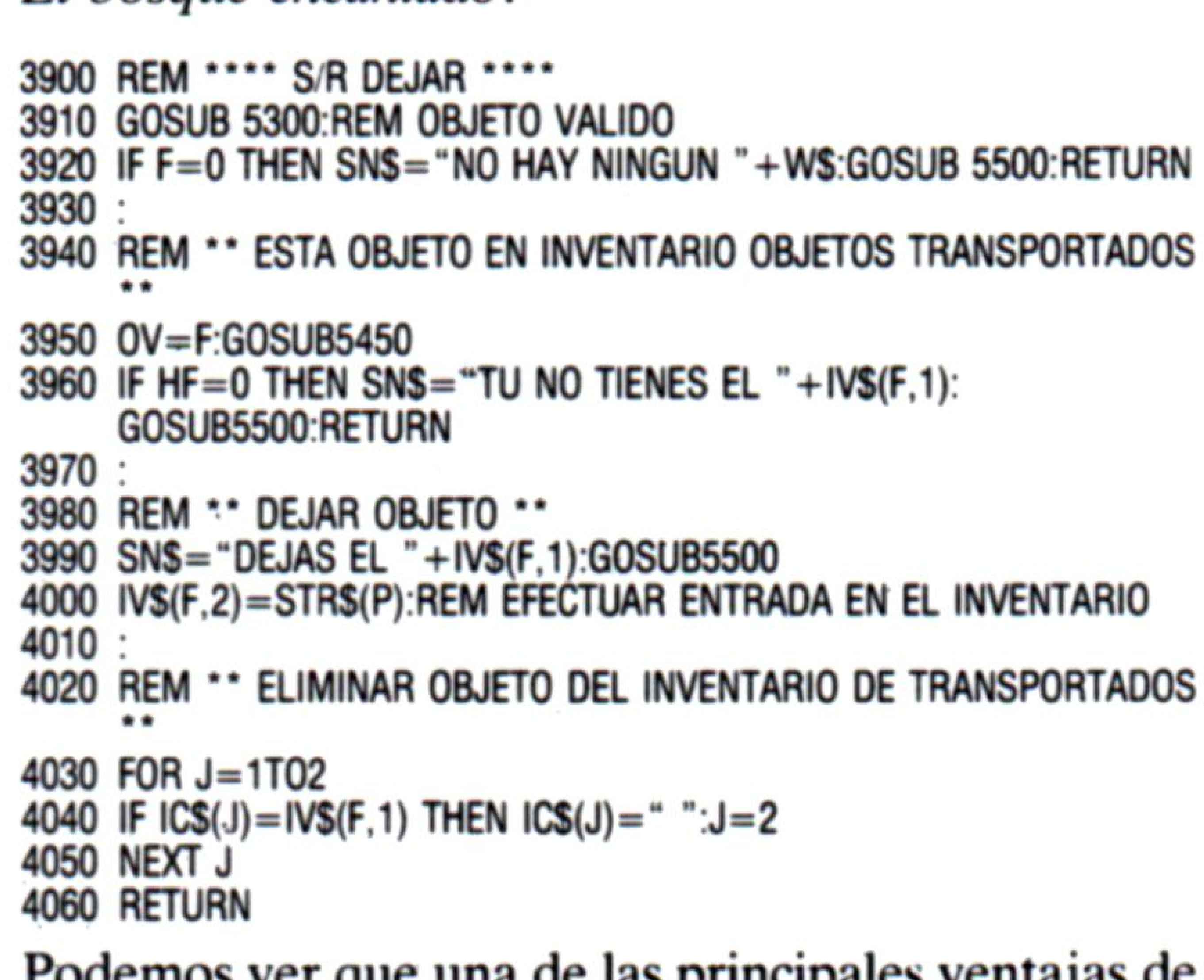
La lógica de la rutina DEJAR es similar a aquella de la rutina RECOGER, pero solamente la validez del objeto y su presencia en el inventario del jugador han de ser comprobados antes de ejecutar la instrucción DEJAR. El inventario principal es entonces actualizado para mostrar un nuevo escenario para el objeto, y el nombre del objeto es eliminado del inventario del jugador

Ian McKinnell

Suponiendo que ambas comparaciones produzcan un resultado favorable, se deben introducir las siguientes modificaciones tanto en el inventario principal como en el del jugador:

1) La posición del objeto a dejar se especificará ahora mediante F. La posición actual, P, debe ahora entrarse en la matriz del inventario principal en la posición IV$(F,2).
2) La descripción del objeto debe eliminarse del inventario personal de los objetos que lleva consigo el jugador, IC$(). La mejor forma de hacerlo es buscar en la matriz hasta hallar el objeto apropiado y sustituirlo por una serie nula.

En el diagrama de flujo vemos la lógica de la rutina DEJAR. Éste es el listado de la rutina para el juego *El bosque encantado*:



```
3900 REM **** S/R DEJAR ****
3910 GOSUB 5300:REM OBJETO VALIDO
3920 IF F=0 THEN SN$="NO HAY NINGUN "+W$:GOSUB 5500:RETURN
3930 :
3940 REM ** ESTA OBJETO EN INVENTARIO OBJETOS TRANSPORTADOS
     **
3950 OV=F:GOSUB5450
3960 IF HF=0 THEN SN$="TU NO TIENES EL "+IV$(F,1):
     GOSUB5500:RETURN
3970 :
3980 REM ** DEJAR OBJETO **
3990 SN$="DEJAS EL "+IV$(F,1):GOSUB5500
4000 IV$(F,2)=STR$(P):REM EFECTUAR ENTRADA EN EL INVENTARIO
4010 :
4020 REM ** ELIMINAR OBJETO DEL INVENTARIO DE TRANSPORTADOS
     **
4030 FOR J=1TO2
4040 IF IC$(J)=IV$(F,1) THEN IC$(J)=" ":J=2
4050 NEXT J
4060 RETURN
```

Podemos ver que una de las principales ventajas de programar por módulos es que se puede acceder a las mismas rutinas para fines diferentes. Utilizando un sistema de banderas o indicadores, las decisiones se pueden tomar dentro de breves subrutinas sobre las cuales no se actúa hasta que el control retorne a la rutina que llamó a la subrutina. Un buen ejemplo del empleo de esta clase de estructura de programa es la prueba de validez que descri-

bimos anteriormente. Esta subrutina es llamada tanto por la rutina RECOGER como por la rutina DEJAR. En cada caso, la subrutina toma una decisión respecto a la validez del objeto incluido en la sentencia de la instrucción. Sin embargo, el flujo del programa no sufre ninguna alteración hasta que se produzca un RETURN a cualquiera de las dos rutinas, RECOGER o DEJAR. Sólo después del retorno la subrutina de la prueba de validez establece el valor de la bandera F, y se produce la bifurcación adecuada. La crítica que se puede hacer a esta técnica es que en realidad estamos comprobando dos veces la misma condición: una para establecer el valor de la bandera y otra para comprobar su valor. Aunque esto es efectivamente así, la mayor flexibilidad y facilidad para la depuración que se consigue mediante el empleo de esta técnica por lo general compensa la contrapartida resultante, es decir, el tiempo de ejecución ligeramente mayor.

Escenarios especiales

Hemos llegado a un punto de nuestro proyecto en el cual tenemos completa la programación del esqueleto del juego; es decir, la programación que permite que el jugador lleve objetos consigo y se desplace a través del mundo de aventuras. Podemos ahora pasar a la siguiente fase de diseño, en la cual consideramos los escenarios "especiales" donde se utilizan los objetos, acechan los peligros y se pone a prueba el ingenio y la destreza del jugador.

Antes de analizar con detalle la programación de las rutinas para uno de los escenarios especiales de *El bosque encantado*, consideremos el código que debemos introducir en el bucle principal del programa para detectar los escenarios especiales. En el listado se han de insertar estas dos líneas:

```
257 GOSUB2700:REM ES ESPECIAL P?
258 IF SF=1 THEN 300:REM SIGUIENTE INSTRUCCION
```

La línea 257 llama a un subrutina para ver si el escenario en curso es especial. De ser así, entonces se pone a uno una "bandera especial", SF. Esto significa que cuando el control retorne finalmente al bucle principal del programa, se podrá evitar la parte del bucle principal que se ocupa de las instrucciones. La subrutina que decide si el escenario actual es especial o no es ésta:

```
2700 REM **** S/R ES ESPECIAL P ****
2705 SF=0:REM RESTAURAR BANDERA ESPECIAL

2716 REM ** OTROS ESCENARIOS ESPECIALES **
2720 ON P GOSUB4590,4690,4790,4590
2730 RETURN
```

Recordará que, cuando diseñamos el mapa original para *El bosque encantado*, numeramos primero los cuatro escenarios especiales. Por consiguiente, podemos simplificar la selección de la subrutina apropiada para cada escenario especial haciendo uso de la instrucción ON...GOSUB. Tal como se ve a partir de la forma en que se la utiliza en la línea 2720, esta instrucción va seguida por una serie de números de línea, y el número de línea apropiado se selecciona de acuerdo al valor de la variable de posición, P. Si, por ejemplo, P es uno, la instrucción bifurcará (GOSUB) a la subrutina que comienza en el primer número de línea de la lista; si P es dos, para la llamada GOSUB se utilizará el segundo número de línea, y así sucesivamente.

Hay cuatro números de línea, uno para cada uno de los escenarios especiales de *El bosque encantado*. Si P fuera mayor de cuatro, entonces el control pasaría simplemente a la línea siguiente. Si cada una de las cuatro subrutinas a las que se puede llamar desde la línea 2720 establece una bandera, SF, se puede detectar el hecho de que P era un escenario especial.

De no llamarse a ninguna rutina, la bandera SF permanecerá establecida en cero, indicando que P no es más que un escenario normal. La instrucción ON...GOSUB es, evidentemente, una alternativa económica a la serie de sentencias IF...THEN que prueban el valor de una variable y bifurcan consecuentemente hacia subrutinas diferentes.

La entrada al túnel

Dos de los escenarios especiales de *El bosque encantado* son las dos entradas a un túnel (escenarios 1 y 4). Para tratar con la simple posibilidad de que el jugador desea entrar en el túnel, necesitamos construir cuidadosamente una rutina que manipule las instrucciones normales y le permita al jugador entrar en el túnel o dar la vuelta y volver al sendero.

```
4590 REM **** S/R ENTRADA AL TUNEL ****
4600 SF=1
4605 SN$="HAS LLEGADO A LA BOCA DE UN LARGO
     TUNEL":GOSUB5500
4610 SN$="PUEDES ENTRAR EN EL TUNEL O RETROCEDER POR EL
     SENDERO":GOSUB5500
4620 :
4625 PRINT:INPUT"INSTRUCCIONES";IS$
4630 GOSUB2500:REM DESCOMPONER INSTRUCCION
4635 IF F=0 THEN 4625:REM INSTRUCCION NO VALIDA
4637 GOSUB3000:REM INSTRUCCIONES NORMALES
4640 IF MF=1 THEN RETURN:REM EL JUGADOR RETROCEDE
4645 IF VF=1 THEN 4625:REM INSTRUCCION OBEDECIDA
4650 REM ** NUEVAS INSTRUCCIONES **
4655 IF VB$="ENTRAR" THEN GOSUB 4700:RETURN
4660 IF VB$="RETROCEDER" AND P=4 THEN MF=1:P=6:RETURN
4665 IF VB$="RETROCEDER" AND P=1 THEN MF=1:P=9:RETURN
4667 SN$="NO COMPRENDO":GOSUB5500:GOTO 4625
```

La rutina empieza por poner SF a uno para indicar el hecho de que se ha llegado a un escenario especial. Después de visualizar un mensaje en la pantalla, de describir la entrada al túnel y las opciones de que dispone el jugador, se solicita una instrucción. Nuevamente, en vez de reinventar la rueda cada vez que deseamos analizar una instrucción, podemos sacar partido de la construcción modular del programa para llamar a las subrutinas "descomponer instrucción" e "instrucción normal" desarrolladas para utilizar en las rutinas RECOGER y DEJAR. Considerando cuidadosamente los estados de las diversas banderas establecidas por estas dos subrutinas, podemos transferir el control al interior de nuestra nueva rutina según se requiera. Analicemos estas banderas de forma individual.

La bandera F establecida por la rutina "descomponer instrucción" indica si la instrucción que se le ha pasado tiene un formato válido. Si la instrucción es de una sola palabra no reconocida por la rutina, entonces F toma el valor cero, en cuyo caso querremos saltar hacia atrás del bucle para tomar otra instrucción.

La bandera MF es establecida por la rutina "instrucción normal" si se requiere la descripción de un escenario; esto sucede cuando se genera una instrucción AVANZAR o MIRAR. Un RETURN al bucle principal del programa permitirá el desplazamiento hasta un nuevo escenario en el primer caso, o que

se describa el mismo escenario y se vuelva a entrar a la rutina especial, en el segundo caso.

La rutina "instrucción normal" establece, asimismo, la bandera VF. Un valor de uno indica que se ha reconocido y obedecido la instrucción, en cuyo caso saltaríamos hacia atrás del bucle para la siguiente instrucción. Si VF <> 1, la instrucción no es ninguna de las normales. Habiéndonos ocupado de las posibilidades de instrucciones normales, podemos añadirle nuevas instrucciones a esta rutina. En este caso, se incluyen dos: ENTRAR, para penetrar en el túnel, y RETROCEDER, para alejarse un escenario desde la entrada al túnel. Puesto que esta rutina está diseñada para funcionar para las dos entradas al túnel, la instrucción RETROCEDER ha de considerar cuál de las dos entradas al túnel está salvando el jugador; ello lo indica P según sea su valor 1 o 4. Por lo tanto, P se puede poner a cero antes de dejar la rutina, de modo que al volver a entrar en el bucle principal del programa se produce un cambio de escenario.

Listados de "Digitaya"

```
1190 GOSUB2670:REM ES ESPECIAL P
1200 IF SF=1 THEN 1250:REM SIGUIENTE BUCLE

2360 REM ** S/R DEJAR **
2370 GOSUB5730:REM ES VALIDO EL OBJETO
2380 IF F=0 THEN PRINT"NO HAY NINGUN";W$:RETURN
2390 :
2400 REM ** LLEVA EL JUGADOR EL OBJETO? **
2410 OV=F:GOSUB5830
2420 IFHF=0THENPRINT"TU NO TIENES EL";IV$(F,1):RETURN
2430 :
2440 REM ** DEJAR EL OBJETO **
2450 SN$="DEJAS EL"+IV$(F,1):GOSUB5880
2460 IV$(F,2)=STR$(P):REM ACTUALIZAR POSICION OBJETO
2470 :
2480 REM ** SUPRIMIR EN LISTA OBJETOS TRANSPORTADOS **
2490 FORJ=1TO4
2500 IF IC$(J)=IV$(F,1)THENIC$(J)=" ":J=4
2510 NEXT J
2520 RETURN

2670 REM **** S/R ES ESPECIAL P ****
2680 SF=0:REM RESTAURAR BANDERA ESPECIAL

2710 ON P GOSUB 2850,2960,3450,3830,4180,4550,5150
2720 RETURN

2850 REM **** S/R TOMA TV ****
2860 SF=1
2870 SN$="HAS ENTRADO EN LA TOMA DEL TELEVISOR Y NO TIENES
     ESCAPATORIA."
2880 SN$=SN$+"ESTAS CONDENADO PARA SIEMPRE A SER UN INVITADO
     DE UN PROGRAMA DE TERTULIA DE LA TV"
2890 GOSUB5880:REM FORMATEAR IMPRESION
2900 PRINT
2910 PRINT"BIENVENIDO AL PROGRAMA......"
2920 FORJ=1TO500:NEXTJ
2930 GOTO 2910
2940 END

3830 REM **** PUERTA PALANCA MANDOS ****
3840 SF=1
3850 SN$="UN USUARIO CON LOS OJOS ENROJECIDOS DISPARA
     REPETIDAMENTE SU LASER CONTRA TI."

3860 GOSUB5880:REM FORMATO
3870 :
3880 REM ** INSTRUCCIONES **
3890 RD=RDN(TI):IF RD>.65THEN 4110:REM ACERTADO
3900 PRINT:INPUT"INSTRUCCIONES";IS$
3910 GOSUB 1700:GOSUB 1900:REM ANALIZAR INSTRUCCION
3920 IFMF=1THENMF=0:PRINT"NO PUEDES
     MOVERTE...TODAVIA":GOTO3880
3930 IFVF=1THEN3880:REM SIGUIENTE INSTRUCCION
3940 IF VB$<>"USAR" THEN PRINT"NO COMPRENDO":GOTO3880
3950 GOSUB5730:REM ES VALIDO EL OBJETO
3960 IFF=0THENPRINT"NO HAY NINGUN";NN$:GOTO3880:REM SIGUIENTE
     INSTRUCCION
3970 :
3980 REM ** ES EL OBJETO ESCUDO LASER **
3990 IF F=3 THEN4020:REM OK
4000 SN$="TU "+IV$(F,1)+"NO SIRVE":GOSUB5880:GOTO3880
4010 :
4020 OV=3:GOSUB5830:REM TIENE CONSIGO EL ESCUDO LASER
4030 IFHF=0THENSN$="TU NO TIENES EL
     "+IV$(3,1):GOSUB5880:GOTO3880
4040 :
4050 REM ** SALVADO **
4060 SN$="UTILIZAS EL ESCUDO LASER PARA PROTEGERTE. UNA RAFAGA
     TE ARROJA"
4070 SN$=SN$+"FUERA DE LA PUERTA PARA PALANCA DE MANDOS Y TE
     HACE ENTRAR OTRA VEZ EN LA MAQUINA."
4080 GOSUB5880:REM FORMATO
4090 P=INT(RND(TI)*40+7):MF=1:RETURN
4100 :
4110 REM ** ACERTADO **
4120 SN$="HAS SIDO ALCANZADO POR EL LASER Y APENAS SI ERES
     CONSCIENTE DE QUE"
4130 SN$=SN$+"TUS ATOMOS HAN QUEDADO DISEMINADOS POR LAS
     CUATRO ESQUINAS"
4140 SN$=SN$+" DEL UNIVERSO"
4150 GOSUB5880:REM FORMATO
4160 END

5150 REM **** S/R PUERTA A LA MEMORIA ****
5160 SF=1
5170 SN$="ERES RECIBIDO POR UN PORTERO QUE TE DICE QUE NO PUEDES
     SER ADMITIDO"
5180 SN$=SN$+"A MENOS QUE DES UNA DIRECCION":
     GOSUB5880
5190 REM ** INSTRUCCIONES **
5200 PRINT:INPUT "INSTRUCCIONES";IS$
5210 GOSUB1700:GOSUB1900:REM ANALIZAR
5220 IF MF=1 THEN RETURN:REM IRSE
5230 IF VF=1 THEN 5200:REM SIGUIENTE INSTRUCCION
5240 IF VB$<>"DAR"THENPRINT"NO COMPRENDO":
     GOTO 5200
5250 :
5260 GOSUB5730:REM ES VALIDO EL OBJETO
5270 IFF=0THENPRINT"NO HAY NINGUN";W$:GOTO5200:REM SIGUIENTE
     INSTRUCCION
5280 :
5290 REM ** ES EL OBJETO UNA DIRECCION **
5300 IF F=1 THEN5330:REM OK
5310 PRINT"EL NECESITA TU DIRECCION":GOTO5200
5320 :
5330 OV=1:GOSUB5830:REM LLEVA CONSIGO LA
     DIRECCION
5340 IF HF=1 THEN 5370
5350 SN$="NO TIENES LA"+IV$(1,1):GOSUB5880:
     GOTO5200
5360 :
5370 REM ** OK PUEDE PASAR **
5380 SN$="EL PORTERO EXAMINA TU DIRECCION Y TE
     PERMITE"
5390 SN$=SN$+"ENTRAR":GOSUB5880
5400 P=40:MF=1:RETURN
```

Complementos al BASIC

Spectrum:

En ambos programas reemplace los siguientes nombres de variables: SN$ por S$, NN$ por R$, IV$(,) por V$(,), IC$ por I$(), IS$ por T$, VB$ por B$.

En el listado de *El bosque encantado* sustituya las siguientes líneas:

```
2720 IF P=1 THEN GOSUB4590
2722 IF P=2 THEN GOSUB4690
2724 IF P=3 THEN GOSUB4790
2726 IF P=4 THEN GOSUB4590
```

En los listados para *Digitaya* reemplace estas líneas:

```
2710 IF P=1 THEN GOSUB2850
2711 IF P=2 THEN GOSUB2960
2712 IF P=3 THEN GOSUB3450
2713 IF P=4 THEN GOSUB3830
2714 IF P=5 THEN GOSUB4180
2715 IF P=6 THEN GOSUB4550
2716 IF P=7 THEN GOSUB5150
3890 LET RD=RND(1)
4090 LET P=INT(RND(1)*40+7)
```

BBC Micro:

En los listados de *Digitaya* reemplace estas líneas:

```
3890 RD=RND(1)
4090 P=RND(40)+7
```

La prueba del tiempo

Examinemos una de las primeras máquinas portátiles compatibles con IBM que aparecieron en el mercado: el Compaq Plus

El lanzamiento en Estados Unidos del IBM Personal Computer confirió cierta respetabilidad al microordenador. Los hombres de negocios, que hasta entonces habían considerado tales dispositivos como poco más que artilugios novedosos, se sintieron felices de convertirse en clientes del célebre gigante de la industria de ordenadores. Dado que las firmas de software comenzaron a crear un ingente número de programas para el IBM, fue inevitable que muchos fabricantes de hardware produjeran ordenadores capaces de ejecutar el software IBM y aprovecharan de este modo la inmensa base de software que se estaba acumulando.

Una de las pioneras en este campo fue la Compaq Computer Corporation, empresa creada específicamente para producir una máquina portátil compatible con IBM: el Compaq Plus. El modelo que vamos a examinar aquí es la versión de 256 Kbytes con unidades de disco gemelas, si bien existen versiones de la máquina con una sola unidad o con un disco rígido fijo de 10 Mbytes.

Teclado y pantalla

"Portátil" tal vez sea un término erróneo para una máquina que pesa 14 kg; ciertamente es difícil transportarla más allá de una corta distancia. Compaq ha reconocido el problema que supone trasladar un peso tan grande y ha proporcionado un asa vinílica acolchada que facilita el transporte.

Como es habitual tratándose de ordenadores portátiles, el teclado del Compaq se engancha en la parte delantera del ordenador. Dado que el asa para transportarlo está instalada en la parte trasera, ello significa que el teclado sirve asimismo como base. No obstante, la máquina es de construcción sólida y el teclado parece aceptar el peso y el consiguiente traqueteo bastante satisfactoriamente. La unidad completa tiene un tamaño aproximado al de una máquina de coser. Una vez desenganchado, el teclado se conecta al ordenador mediante un cable en espiral. El fabricante afirma que esto permite situar el teclado de manera que se pueda hallar la posición de trabajo más cómoda. Sin embargo, si se separa de la unidad más de unas ocho pulgadas, comienza a ser estirado por la espiral del cable, limitando por lo tanto la distancia entre la pantalla y el usuario. Esto se podría haber solucionado con un cable más delgado.

El teclado, al igual que el propio ordenador, posee patas plegables que permiten colocarlo en ángulo para trabajar con comodidad. Tal como uno esperaría de un ordenador de oficina como el Compaq Plus, las teclas son fiables y fáciles de usar. La disposición del teclado es idéntica a la del IBM-PC, con 10 teclas de función junto al lado izquierdo de las teclas de máquina de escribir, y un teclado numérico en el lado derecho. La ventaja obvia de esta disposición es que los usuarios que estén habituados a la del IBM podrán utilizar el teclado del Compaq instantáneamente sin tener que aprenderse la nueva posición de las teclas. La desventaja reside en que el teclado del IBM no es el ideal y, por consiguiente, se reproducen los mismos problemas de diseño. La tecla Enter del IBM es del mismo tamaño que las otras teclas, por lo cual hallarla resulta difícil. Lo que quizá sea más grave para quienes escriben al tacto es que la tecla Shift no está en el lugar habitual, es decir, debajo de la tecla A. Esta posición la ocupa, en cambio, la barra invertida, lo que significa que el mecanógrafo al tacto se pasará, al menos en principio, pulsando continuamente la barra invertida en vez de Shift. Estos problemas se reproducen en el Compaq.

Cuando se ejecuta el MS-DOS (la versión Microsoft del PC-DOS), las teclas de función situadas a la izquierda del teclado actúan como medios auxiliares a la edición. Estas funciones varían a tenor del programa de aplicaciones que se esté ejecutando. Con el BASIC, por ejemplo, se convierten en teclas de entrada de palabras clave individuales, tales como LOAD, SAVE y LIST. Las máquinas MSX utilizan un sistema similar para las teclas de función.


Chris Stevens

"Compaqtando"
Las líneas suaves y bien definidas de la carcasa del Compaq se repiten en el sencillo panel frontal y en el moderado facsímil del teclado del IBM-PC. A ambos lados de la carcasa, unas tapas dan acceso a las puertas y a la fuente de alimentación eléctrica, como asimismo a un espacio de almacenamiento para el cable de potencia y el enchufe de la red, ¡un punto importante y en ocasiones menospreciado!

El teclado numérico situado a la derecha del teclado tiene una doble función. En operación normal, se lo puede emplear para desplazar el cursor a través de la pantalla, pero si se pulsa la tecla Num Lock se lo puede utilizar a modo de calculadora.

La pantalla es un monitor verde estándar de 17,7 × 13,3 cm, con una resolución de 80 × 25 caracteres. El texto se lee fácilmente y a la derecha de la pantalla hay un control de brillo. En el lado izquierdo están las unidades gemelas de disco flexible de 5 1/4 pulgadas. Debido a que la configuración estándar del Compaq Plus es una única unidad de disco, el disco maestro del sistema MS-DOS dará por sentado que es éste el caso y pedirá que todos los discos se coloquen en la unidad A; esto puede ser una molestia cuando se está copiando uno, porque el usuario tiene que intercambiar los discos continuamente. Sin embargo, el MS-DOS se puede configurar para hacer un uso completo de las dos unidades. Las unidades propiamente dichas parecen ser más silenciosas que sus equivalentes IBM.

Conexiones al exterior

Debajo de un panel al lado derecho de la máquina están las puertas de interface. El Compaq Plus está equipado con una puerta para impresora en paralelo tipo Centronics, una interface RGB y una puerta RF. También se proporcionan tres ranuras de ampliación, permitiendo instalar placas con conectores compatibles con IBM. Las ampliaciones típicas son placas de memoria extra, una tarjeta de color VDU o un modem que le permita al ordenador comunicarse a través de la red telefónica. Debajo de otro panel situado en el lado izquierdo está la entrada de potencia (un Eurosocket estándar de tres patillas), encima de la cual está el interruptor on/off. A uno de los lados está el ventilador que mantiene frío el interior de la máquina. Dentro del ordenador, el sistema de circuitos está protegido por una carcasa metálica (motivo, en parte, del peso de la máquina); ésta no sólo ofrece protección contra manejos bruscos, sino que también hace de escudo contra las interferencias de radio que pudieran perturbar el proceso. El metal actúa asimismo como disipador.

El ordenador viene acompañado de tres manuales: una guía de funcionamiento y guías de referencia de MS-DOS y de BASIC. Insólitamente, tratándose de este tipo de máquina, las guías están encuadernadas en forma de libro y no son las usuales carpetas de anillas. Los libros vienen junto con los discos del sistema en carpetas plásticas imitación de ante. De los tres manuales, sólo la guía para operadores tiene un enfoque didáctico. De modo que quien desee documentarse sobre los diversos pasos de la utilización del BASIC o del MS-DOS hará mejor en adquirir otros textos de enseñanza.

Como una de las más antiguas máquinas compatibles con IBM, el Compaq, al igual que el propio IBM-PC, utiliza el procesador Intel 8088 en vez del chip 8086 más avanzado que emplean algunos de sus competidores más recientes, como el Olivetti M25. Si bien el 8088 es un procesador de 16 bits con un bus de direcciones de 20 bits, posee un bus de datos de 8 bits, a diferencia del 8086, que posee un bus de 16 bits. Esto significa que el Compaq es mucho más lento para recuperar datos que sus rivales que poseen el 8086, aunque, por supuesto, opera a la misma velocidad que el PC.

Respecto a la cuestión, sumamente importante, de la compatibilidad con el software IBM, el Compaq Plus se encuentra en una posición muy airosa. Hasta un programa tan difícil como el *Lotus 1-2-3* se puede ejecutar en la máquina. Sin embargo, quizá este hecho no resulte sorprendente si se tiene en cuenta que el presidente de Compaq también es miembro de la junta directiva de Lotus Software. Una de las pocas cosas que el Compaq Plus no puede ejecutar es el disco de diagnóstico IBM; pero, puesto que ese programa interroga directamente a la ROM del sistema de E/S de BASIC (BIOS: BASIC Input-Output System), ello tampoco ocasiona inconvenientes.

El Compaq Plus es, en definitiva, una de las máquinas compatibles con IBM más fiables que existen en el mercado, con una trayectoria ya probada.

En el aspecto negativo, la máquina está empezando a dar muestras de su edad, y no sólo por el chip 8088, ya anticuado. Está empezando a dar la sensación de que están contados los días de los micros "transportables" de más de 10 kg. Con el creciente y continuo perfeccionamiento de los ordenadores "de regazo", la aparición de una máquina de este tipo compatible con el IBM es sólo cuestión de tiempo.

Costo-prestaciones

Los principales argumentos de venta del Compaq Plus son su compatibilidad con IBM y, por consiguiente, el acceso a la enorme base de software IBM. Aquí se puede apreciar el costo de un sistema IBM-PC equiparable a las especificaciones básicas del Compaq Plus:

Configuración	Compaq Plus	IBM-PC
Precio básico	£ 2 524	£ 1 310
Discos gemelos 360 K	estándar	341
Pantalla color alta res.	estándar*	300
Tarjeta salida monitor	estándar	208
Tarjeta gráficos color	estándar	223
128 K de RAM	estándar	86
Sistema operativo y BASIC basado en disco	estándar	61
Teclado	estándar	213
Total	2 524	2 742

* La pantalla incorporada es monocromática de alta resolución, pero el Compaq posee una salida en color RGB como estándar.



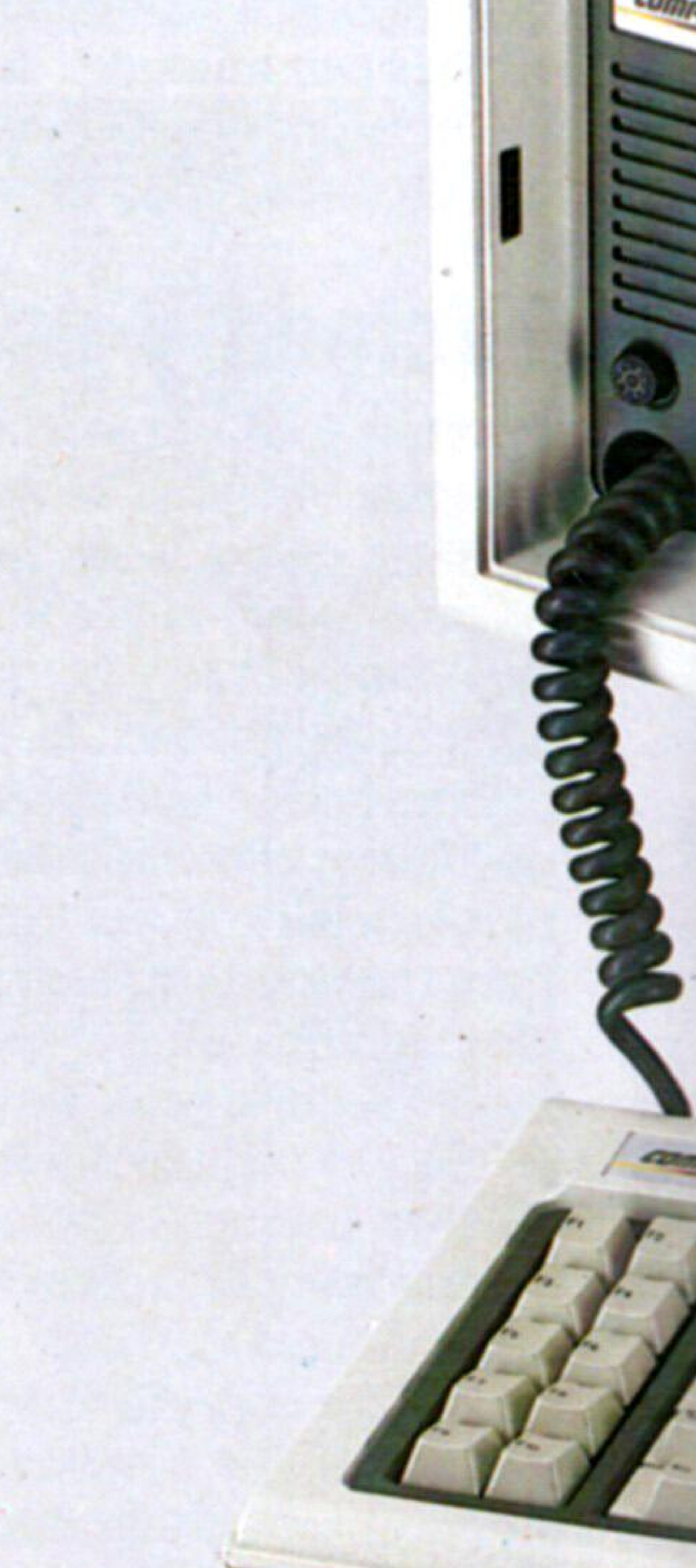
Chris Stevens

Placa impresora
Esta placa con una interface para impresora en paralelo permite conectar la máquina a una impresora de tipo Centronics

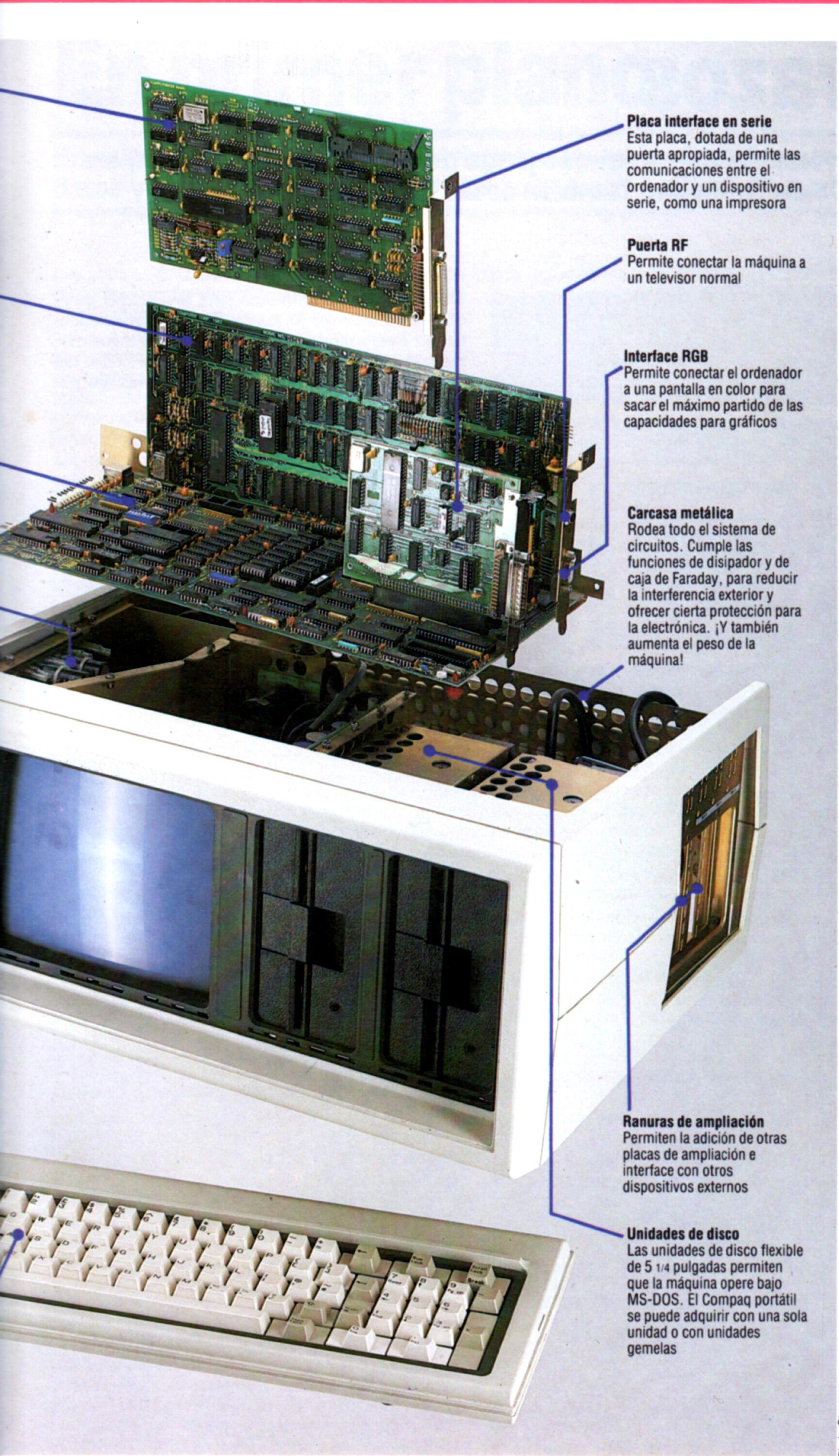
Placa VDU
Se encaja en una de las puertas para ampliación y proporciona el sistema de circuitos necesario para activar un televisor o pantalla externa

Placa de circuitos principal
Contiene los 256 K de RAM, el procesador 8088 y los chips de entrada/salida. Asimismo hay disponibles ranuras libres para chips extras tales como el procesador de matemáticas 8087

Fuente de alimentación eléctrica
Está instalada aquí, con un ventilador por detrás para mantener refrigerado el interior de la máquina



Teclado estilo IBM
Es idéntico a la versión IBM. Observe las teclas de función a la izquierda y el teclado numérico a la derecha



Placa interface en serie
Esta placa, dotada de una puerta apropiada, permite las comunicaciones entre el ordenador y un dispositivo en serie, como una impresora

Puerta RF
Permite conectar la máquina a un televisor normal

Interface RGB
Permite conectar el ordenador a una pantalla en color para sacar el máximo partido de las capacidades para gráficos

Carcasa metálica
Rodea todo el sistema de circuitos. Cumple las funciones de disipador y de caja de Faraday, para reducir la interferencia exterior y ofrecer cierta protección para la electrónica. ¡Y también aumenta el peso de la máquina!

Ranuras de ampliación
Permiten la adición de otras placas de ampliación e interface con otros dispositivos externos

Unidades de disco
Las unidades de disco flexible de 5 1/4 pulgadas permiten que la máquina opere bajo MS-DOS. El Compaq portátil se puede adquirir con una sola unidad o con unidades gemelas

Chris Stevens

COMPAQ PLUS

DIMENSIONES
480 × 400× 200 mm

CPU
Intel 8088, 4,7 MHz

MEMORIA
256 K de RAM, ampliables a 640 K

PANTALLA
Textos: 25 filas de 80 columnas. Gráficos: 640 × 200 pixels

INTERFACES
Impresora en paralelo tipo Centronics, RGB y un enchufe hembra RF. También hay ranuras de ampliación que permiten instalar otras placas de interface

LENGUAJES
BASIC cargado desde el disco maestro del sistema

TECLADO
47 teclas tipo máquina de escribir, 14 teclas de control, 10 teclas de función y 10 teclas de función para cálculo

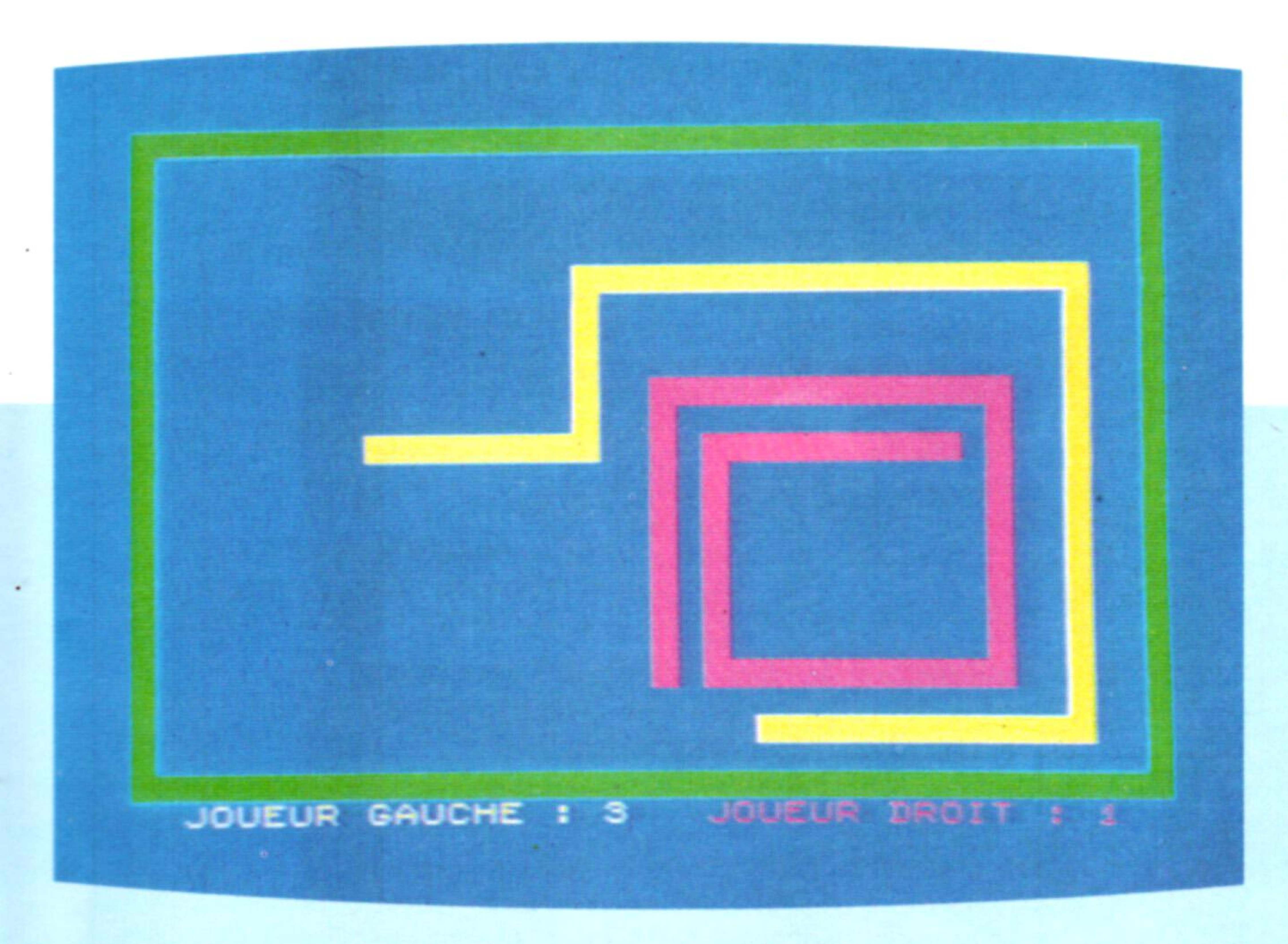
DOCUMENTACION
Se suministran tres manuales: una guía de funcionamiento, una guía de referencia del MS-DOS y una guía de referencia de BASIC. De ellos sólo la guía de funcionamiento lleva al lector paso a paso a través de los procedimientos. Los principiantes deberán adquirir otros manuales con fines didácticos

VENTAJAS
La construcción de la máquina es sólida y su compatibilidad con el IBM-PC está demostrada

DESVENTAJAS
El peso de la máquina la coloca más bien en la categoría de "transportable". Asimismo, el Compaq Plus está en peligro de ser superado por la tecnología más reciente

Trazos

Esta versión de este famoso juego de acción ha sido escrita para el MO5 de Thomson. También está disponible para el TO 7



Dos jugadores se enfrentan para dividirse el espacio vital. Cada uno de ellos debe esforzarse, mientras va desplazándose, por no cortar en ningún momento su trazado o el de su adversario ni salirse del rectángulo dibujado en la pantalla. Utilice las palancas de mando o las teclas siguientes:

Jugador de la derecha: P, L, M y @.
Jugador de la izquierda: Z, Q, S y W.

```
10 REM **********
20 REM * TRAZOS *
30 REM **********
40 CLEAR ,,1
50 GOSUB 750
60 ON JK GOTO 130
70 D$=INKEY$
80 HB=(D$="L")-(D$="M")
90 VB=(D$="P")-(D$="@")
100 HA=(D$="Q")-(D$="S")
110 VA=(D$="Z")-(D$="W")
120 GOTO 170
130 HA=(STICK(0)=7)-(STICK(0)=3)
140 VA=(STICK(0)=1)-(STICK(0)=5)
150 HB=(STICK(1)=7)-(STICK(1)=3)
160 VB=(STICK(1)=1)-(STICK(1)=5)
170 IF HA<>0 THEN H1=HA:V1=0
180 IF VA<>0 THEN V1=VA:H1=0
190 IF HB<>0 THEN H2=HB:V2=0
200 IF VB<>0 THEN V2=VB:H2=0
210 X1=X1+H1
220 Y1=Y1+V1
230 IF SCREEN(X1,Y1)=127 THEN 350
240 LOCATE X1,Y1
250 COLOR 3
260 PRINT N$;
270 X2=X2+H2
280 Y2=Y2+V2
290 IF SCREEN(X2,Y2)=127 THEN 410
300 LOCATE X2,Y2
310 COLOR 5
320 PRINT N$;
330 BEEP
340 GOTO 60
350 F2=F2+1
360 GOSUB 690
370 IF F2=10 THEN 470
380 IF INKEY$<>" " THEN 380
390 GOSUB 870
400 GOTO 60
410 F1=F1+1
420 GOSUB 690
430 IF F1=10 THEN 540
440 IF INKEY$<>" " THEN 440
450 GOSUB 870
460 GOTO 60
470 CLS
480 COLOR 1,7
490 LOCATE 9,5
500 PRINT "GANA JUGADOR DERECHA";
510 LOCATE 15,10
520 PRINT F2;"A";F1;
530 GOTO 600
540 CLS
550 COLOR 1,7
560 LOCATE 9,5
570 PRINT "GANA JUGADOR IZQUIERDA";
580 LOCATE 15,10
590 PRINT F1;"A";F2;
600 LOCATE 14,15
610 PRINT "OTRA ?";
620 IF INKEY$<>" " THEN 620
630 D$=INKEY$
640 IF D$=" " THEN 630
650 IF D$<>"N" THEN RUN
660 CLS
670 SCREEN 4,6,6
680 END
690 FOR I=1 TO 5
700 BEEP
710 FOR J=1 TO 100
720 NEXT J
730 NEXT I
740 RETURN
750 CLS
760 SCREEN 2,4,4
770 DEFINT A-Z
780 ATTRB 1,1
790 LOCATE 6,10,0
800 PRINT "PALANCAS DE MANDO ?";
810 R$=INKEY$
820 IF R$=" " THEN 810
830 JK=-(R$="O")
840 ATTRB 0,0
850 DEFGR$(0)=255,255,255,255,255,255,255,255
860 N$=GR$(0)
870 CLS
880 COLOR 2
890 FOR X1=0 TO 39
900 LOCATE X1,0:PRINT N$;
910 LOCATE X1,23:PRINT N$;
920 NEXT X1
930 FOR Y1=1 TO 22
940 LOCATE 0,Y1:PRINT N$;
950 LOCATE 39,Y1:PRINT N$;
960 NEXT Y1
970 LOCATE 2,24:COLOR 3
980 PRINT "JUGADOR IZQUIERDA :";F1;
990 LOCATE 22,24:COLOR 5
1000 PRINT "JUGADOR DERECHA :";F2;
1010 X1=8:Y1=11:X2=32:Y2=11:H1=1:V1=0
1020 H2=-1:V2=0:S1=0:S2=0:RETURN
```



En primer plano

En esta ocasión estudiaremos distintos tipos de visualización de datos y desarrollaremos una rutina para crear gráficos de barras

Los gráficos de barras son un sencillo método gráfico de representar ciertos tipos de datos numéricos. El objetivo de un gráfico de barras es ayudar a entender a simple vista un conjunto de cifras sin tener que examinarlas con mayor detalle. Existen muchos programas para gráficos de gestión para trazar gráficos de barras, gráficos de tarta e histogramas, y estos programas a menudo están enlazados con hojas electrónicas de modo que se puedan visualizar los valores calculados en las mismas.

Nosotros, por el momento, no seremos tan ambiciosos, sino que comenzaremos por analizar cómo dibujar un gráfico de barras utilizando el LOGO. La versión del programa que vamos a considerar es para el Commodore 64; para ejecutar el programa en otras máquinas, el lector habrá de remitirse a los *Complementos al* LOGO, que incluyen los cambios necesarios en cada caso.

El proceso del dibujo de un gráfico de barras se divide en tres etapas:

a) tomar la entrada;
b) hallar el valor mayor (esto es necesario para que podamos colocar a escala las columnas de modo que quepan en la pantalla);
c) dibujar el gráfico de barras, a escala apropiada, y añadir luego las etiquetas.

El procedimiento de nivel superior se denomina GRAFICOBARRAS.INIC establece el valor de una cantidad de constantes que necesita el programa. Reuniéndolas todas en un mismo lugar se simplifica la labor de introducir modificaciones. COLORES contiene la lista de colores a utilizar para las columnas; si no le agrada nuestro esquema de color, ¡cámbielo! Tal como está, el programa imprimirá un máximo de 15 columnas, de modo que sólo se necesitarán 15 colores como máximo.

Entrada y cálculo

La entrada que se requiere consta de dos datos para cada una de las barras del gráfico; primero, el título a imprimir al pie de la barra y, segundo, la cantidad o el valor de la barra, o sea, su altura. En nuestras rutinas de entrada hemos incorporado algunas sencillas comprobaciones de verificación para asegurar que la entrada sea sensata. TOMAR.ENTRADA divide en dos partes la tarea de obtener la entrada, tomando los títulos para cada columna e insertando los valores correspondientes. Cuando acabe de entrar los datos, digite FIN como el siguiente "título".

TOMAR.TITULO acepta el título; rechazará una entrada en blanco pero aceptará cualquier otro valor. TOMAR.CANTIDAD aceptará como entrada sólo un número; cualquier otra entrada hará que el programa solicite que los datos sean reentrados. Cuando TOMAR.ENTRADA posee un par nombre/número válido, éste se añade al final de una lista de elementos de datos. Los elementos se deben entrar por el orden en el que uno desea que se tracen en la pantalla, de izquierda a derecha.

CALCULAR utiliza TOMAR.MAX para hallar el valor mayor y luego emplea éste para establecer una escala. Una regla común es que la altura del gráfico de barras debe ser de aproximadamente las tres cuartas partes de la anchura.

DIBUJAR.GRAFICO calcula la anchura de cada barra del gráfico, dibuja un eje hacia arriba de la pantalla, colorea y etiqueta las barras.

La anchura se calcula como un múltiplo de ocho. Ello se hace con el fin de evitar algunos problemas originados por la forma en que el Commodore 64 visualiza los colores, porque en la modalidad para gráficos normal del LOGO no se pueden tener dos colores en el mismo bloque de 8 × 8 pixels. DIBUJAR.EJE dibuja la línea hacia arriba de la pantalla y marca en ella el mayor de los valores de los datos; esto nos proporciona una forma sencilla de estimar los valores de las columnas.

Debemos retroceder desde la línea del eje antes de imprimir este número junto a la máscara que indica el valor mayor. El espacio a retroceder, de modo que el número no se imprima sobre la línea, depende de cuántos dígitos tenga el número. Este problema se resuelve fácilmente, porque el LOGO trata los números como si fueran palabras, de manera que podemos utilizar CONTAR para determinar la longitud del número, y luego emplearlo para determinar cuánto se ha de retroceder.

ESCRIBIR imprime un mensaje en la pantalla para gráficos. Toma tres entradas: las distancias de pasos X e Y para cada carácter y el nombre a escribir. Se vale de una primitiva, STAMPCHAR, que imprime un carácter en la pantalla para gráficos, en la posición de la tortuga.

DIBUJAR.GRAFICO1 lleva a cabo la tarea de dibujar las barras. Toma cada elemento de uno en uno, selecciona el siguiente color a utilizar, somete a escala la altura y le pasa el verdadero trabajo a RELLENAR, que simplemente recorre la barra de arriba abajo rellenándola. ETIQUETA emplea ESCRIBIR para escribir las etiquetas verticalmente hacia abajo de la pantalla (las muy largas se desplazarán y aparecerán en la parte de arriba de la pantalla).

Gráficos de tarta

Los gráficos de tarta son otra forma muy común de representación gráfica. Si desea escribir un programa para dibujar un gráfico de tarta, he aquí algunas pistas:

- Los datos se obtienen exactamente de la misma forma que en el caso del gráfico de barras.
- La sección de cálculos consiste en totalizar los

números y determinar a partir de este dato qué porción les corresponderá de los 360° que constituyen la "tarta".

- Dibujar y rellenar los trozos de la tarta es bastante sencillo, pero en el Commodore 64, al menos, usted tendrá algunos problemas debido a la forma en que los colores se invaden entre sí. Es una buena idea aplicar la modalidad de "color doble"; tan sólo utilice DOUBLECOLOR en lugar de DRAW en el procedimiento que dibuja el gráfico. Si aun así tuviera problemas, deje un "agujero" de 10 unidades en el centro del gráfico de tarta.

El etiquetado se puede realizar de la misma forma que para el gráfico de barras, si bien podrían plantearse problemas al posicionar la tortuga antes de efectuar la escritura.

Después de que logre este gráfico, ¿por qué no tratar de escribir un programa que dibuje tanto un gráfico de barras como uno de tarta para los mismos datos, uno junto al otro?

Complementos al LOGO

Muchas versiones de LOGO no poseen un equivalente para STAMPCHAR, lo que hace que imprimir caracteres en la pantalla para gráficos resulte muy difícil.
Para las diferentes máquinas será necesario alterar los números de color y de tamaño del gráfico de barras. Todos estos detalles están reunidos en INIC, de modo que pueden tratarse todos juntos de una sola vez.

Para todas las versiones LCSI:
Utilice TYPE por PRINT1
Utilice EMPTYP por EMPTY?
Tal vez necesite emplear EQUALP en lugar del signo de igualdad (=)
SETXY debe ir seguida de una lista
IF posee una sintaxis diferente; por ej.:

```
IF EMPTYP :LISTADATOS [STOP]
```

Gráficos

Tanto los gráficos de barras como los gráficos de tarta constituyen unas formas muy útiles para visualizar información, si bien el LOGO no es ideal para estos fines debido a su poca velocidad

ENTRADA DE DATOS

	Londres	Nueva York
ENE	53	94
FEB	40	97
MAR	37	91
ABR	38	81
MAY	46	81
JUN	46	84
JUL	56	107
AGO	59	109
SEP	50	86
OCT	57	89
NOV	64	76
DIC	48	91

Datos de precipitaciones (en milímetros)



Programa para el gráfico de barras

```
TO GRAFICO.BARRAS
  INIC NODRAW
  TOMAR.ENTRADA CALCULAR
  DIBUJAR.GRAFICO
END

TO INIC
  MAKE "COLORES [0 1 2 5 6 0 1 2 5 6 0 1 2 5 6]
  MAKE "FONDO 11
  MAKE "DATOS []
  MAKE "ALTURA 160
  MAKE "ANCHURA 190
  MAKE "XORIGEN (−104)
  MAKE "YORIGEN (−40)
END

TO TOMAR.ENTRADA
  TOMAR.TITULO
  IF FIRST :TITULO = "FIN THEN STOP
  TOMAR.CANTIDAD
  MAKE "DATOS LPUT SENTENCE :TITULO
     :CANTIDAD :DATOS
  TOMAR.ENTRADA
END

TO TOMAR.TITULO
  PRINT "
  PRINT [(ENTRE FIN PARA TERMINAR)]
  (PRINT1 "TITULO: ' ')
  MAKE "TITULO REQUEST
  IF EMPTY? :TITULO THEN TOMAR.TITULO
END

TO TOMAR.CANTIDAD
  (PRINT1 "CANTIDAD:"' ')
  MAKE "CANTIDAD REQUEST
  IF EMPTY? :CANTIDAD THEN TOMAR.CANTIDAD
     ELSE IF NOT NUMBER?
  FIRST :CANTIDAD THEN TOMAR.CANTIDAD
END

TO CALCULAR
  MAKE "MAX 0
  TOMAR.MAX :DATOS
  MAKE "ESCALA :ALTURA/:MAX
END

TO TOMAR.MAX :LISTADATOS
  IF EMPTY? :LISTADATOS THEN STOP
  IF LAST FIRST :LISTADATOS > :MAX THEN
     MAKE "MAX LAST FIRST :LISTADATOS
  TOMAR.MAX BUTFIRST :LISTADATOS
END

TO DIBUJAR.GRAFICO
  DRAW BACKGROUND :FONDO
  HIDETURTLE FULLSCREEN
  MAKE "ANCHURA ((ROUND ((:ANCHURA/
     COUNT :DATOS)/8))*8)−8
  PENUP SETXY :XORIGEN :YORIGEN
  DIBUJAR.EJE
  DIBUJAR.GRAFICO1 :COLORES :DATOS
  ETIQUETA :DATOS
END

TO DIBUJAR.EJE
  PENDOWN
  SETY YCOR+:ALTURA+10
  SETY YCOR−10
  SETX XCOR+4
  SETX XCOR−8
  PENUP
  SETX XCOR−(8*COUNT :MAX)
  ESCRIBIR 8 0:MAX
  SETX XCOR+4
  SETY :ORIGEN
END

TO ESCRIBIR :XINC :YINC :NOMBRE
  IF EMPTY? :NOMBRE THEN STOP
  STAMPCHAR FIRST :NOMBRE
  SETXY XCOR+:XINC YCOR+:YINC
  ESCRIBIR :XINC :YINC BUTFIRST :NOMBRE
END

TO DIBUJAR.GRAFICO1 :COLORESLAPIZ
   :LISTADATOS
  IF EMPTY? :LISTADATOS THEN STOP
  SETX XCOR+8
  PENCOLOR FIRST :COLORESLAPIZ
  PENDOWN
  RELLENAR ((LAST FIRST :LISTADATOS)*
     :ESCALA) :ANCHURA
  PENUP
  DIBUJAR.GRAFICO1 BUTFIRST
     :COLORESLAPIZ BUTFIRST :LISTADATOS
END

TO RELLENAR :LARGO :ANCHO
  IF :ANCHO=0 THEN STOP
  FORWARD :LARGO RIGHT 90
  FORWARD 1 LEFT 90
  BACK :LARGO RIGHT 90
  FORWARD 1 LEFT 90
  RELLENAR :LARGO :ANCHO−2
END

TO ETIQUETA :LISTADATOS
  ETIQUETA1 (:XORIGEN+8+ :ANCHURA/2)
     (:YORIGEN−10)
  :LISTADATOS
END

TO ETIQUETA1 :X :Y :LISTADATOS
  IF EMPTY? :LISTADATOS THEN STOP
  SETXY :X :Y
  ESCRIBIR 0(−10) FIRST FIRST :LISTADATOS
  ETIQUETA1 ( :X+ :ANCHURA+8)
     :Y BUTFIRST :LISTADATOS
END
```



Cuatro sensores

Ahora montaremos los sensores en el robot y escribiremos un programa

El control bidireccional de los motores paso a paso exige cuatro de las ocho líneas de datos de la puerta para el usuario de que disponemos. Ello nos deja cuatro líneas que se pueden utilizar para llevar información desde los sensores hasta el ordenador. Para conferirle más flexibilidad de operación a nuestro robot aplicaremos un "sistema de parches" para permitir la conexión de diferentes permutaciones de los sensores a las cuatro líneas de entrada disponibles. De momento, conectaremos cuatro sensores de microinterruptor y luego instalaremos dos sensores luminosos. Para seleccionar cualquier combinación de estos sensores (p. ej., dos sensores de microinterruptor y dos sensores luminosos), cablearemos cada sensor a un conector en la tapa del robot. Asimismo, se conectarán cuatro conectores a las líneas de datos (de D4 a D7) del enchufe D. Por consiguiente, podemos conectar el sensor apropiado para cualquiera de las cuatro líneas de datos mediante el empleo de un trozo de cable que se enchufe por un extremo en el conector del sensor y, por el otro, en uno de los conectores de las líneas de datos.

Para comprobar la construcción y el cableado de los cuatro microinterruptores podemos escribir un programa muy simple que explore los cuatro bits superiores del registro de datos y visualice los valores decimales de los bits enviados a 0. Ejecute el programa con los cuatro sensores conectados, vía el sistema de conectores con parches, a las líneas de datos de D4 a D7. El cierre de cualquiera de los microinterruptores provocará un cambio en la visualización en pantalla.

INSTALACIÓN

El tipo de microinterruptores que se menciona en la lista de componentes se puede adquirir en la mayoría de tiendas de componentes. En primer lugar, efectúe los cortes necesarios en la tapa de la carcasa del robot. Las ocho ranuras se deben cortar de modo que apenas quepan a través de ellas los conectores de abajo de cada microinterruptor. Los 10 agujeros de 5 mm de diámetro son para los conectores de parche, los cuales se pueden instalar en esta etapa, montando los cuatro conectores rojos en línea lo más cerca posible del enchufe D. Para este proyecto los microinterruptores se han de adaptar ligeramente. La palanca larga del microinterruptor debe inclinarse cuidadosamente, utilizando un par de pinzas, de modo que se forme un ángulo recto. Debe tenerse cuidado en asegurar que la inclinación no quede demasiado cerca de la caja del microinterruptor, porque en ese caso la palanca no cerrará el microinterruptor una vez que éste esté montado en la tapa. De la parte trasera de éste sobresalen dos conectores. De éstos, el de arriba es el conector NC ("normalmente cerrado"). Dado que en nuestro proyecto no lo vamos a utilizar, quizá sea mejor quitarlo, ya sea quebrándolo o cortándolo con una pequeña sierra para metales. El conector de abajo, el NO ("normalmente abierto") sí se utiliza y se lo debe inclinar en ángulo recto junto a la carcasa del interruptor. Ello asegurará que pase, junto con el COM ("conector común"), a través de las ranuras cortadas a tal fin en la tapa de la carcasa del robot. Una vez efectuadas estas modificaciones en cada uno de los cuatro microinterruptores, éstos ya se pueden montar, tal como se indica, en cada esquina de la tapa. Los interruptores deben montarse de modo que las palancas activadoras se cuelguen sobre la parte delantera y trasera de la carcasa del robot, y deben ser encolados en su sitio utilizando un adhesivo adecuado ("supercola" o Cyanoacrylate)

```
10 REM **** PRUEBA SENSORES BBC ****
20 MODE 7:OP=-1:RDD=&FE62:REGDAT=&FE60:?RDD=15
30 PE=240-(?REGDAT AND 240):IF PE=OP THEN 30
40 CLS:PRINT PE:OP=PE:GOTO 30
```

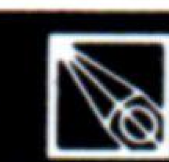
```
10 REM **** PRUEBA SENSORES CBM ****
20 OP=-1:RDD=56579:REGDAT=56577:POKE RDD,15
30 PE=240-(PEEK(REGDAT)AND 240):IF PE=OP THEN 30
40 PRINT CHR$(147):PRINT PE:OP=PE:GOTO 30
```

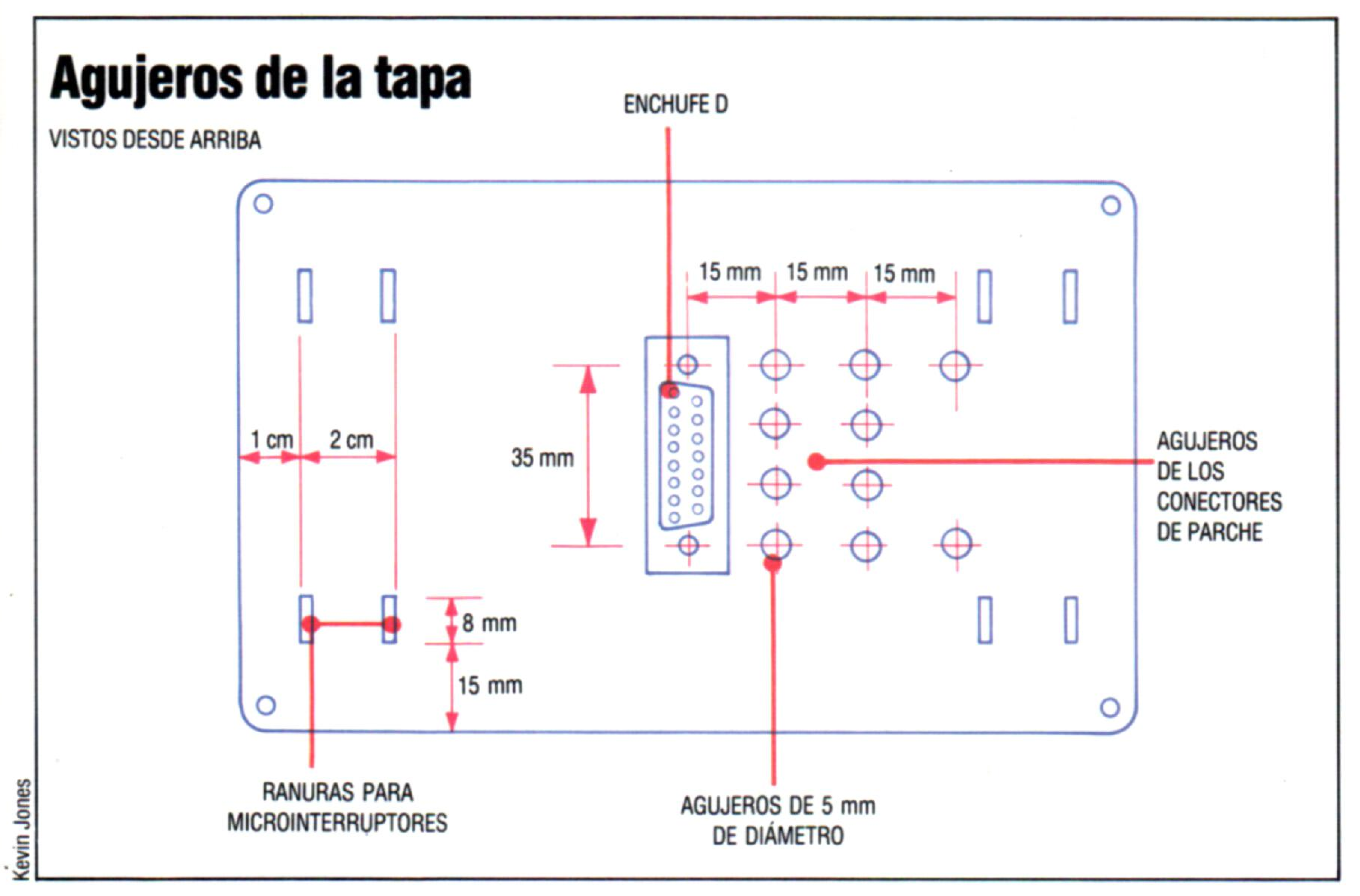
El circuito

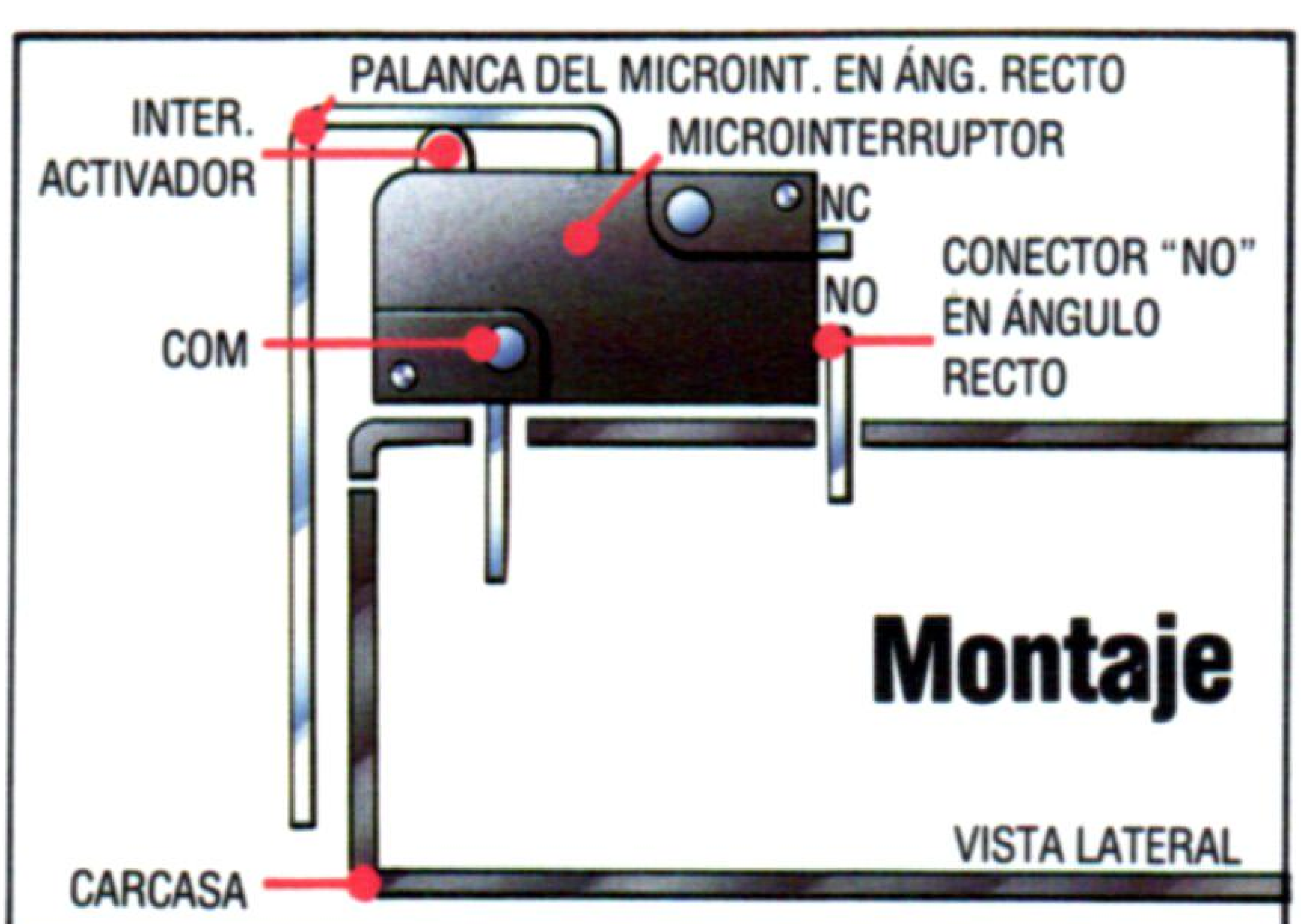
CONECTORES DE PARCHE
MICROINTERRUPTORES
D7
D6
D5
D4
D3
D2
D1
D0
CONTROL DEL MOTOR
CONECTORES DE PARCHE DE LOS MICROINTERRUPTORES

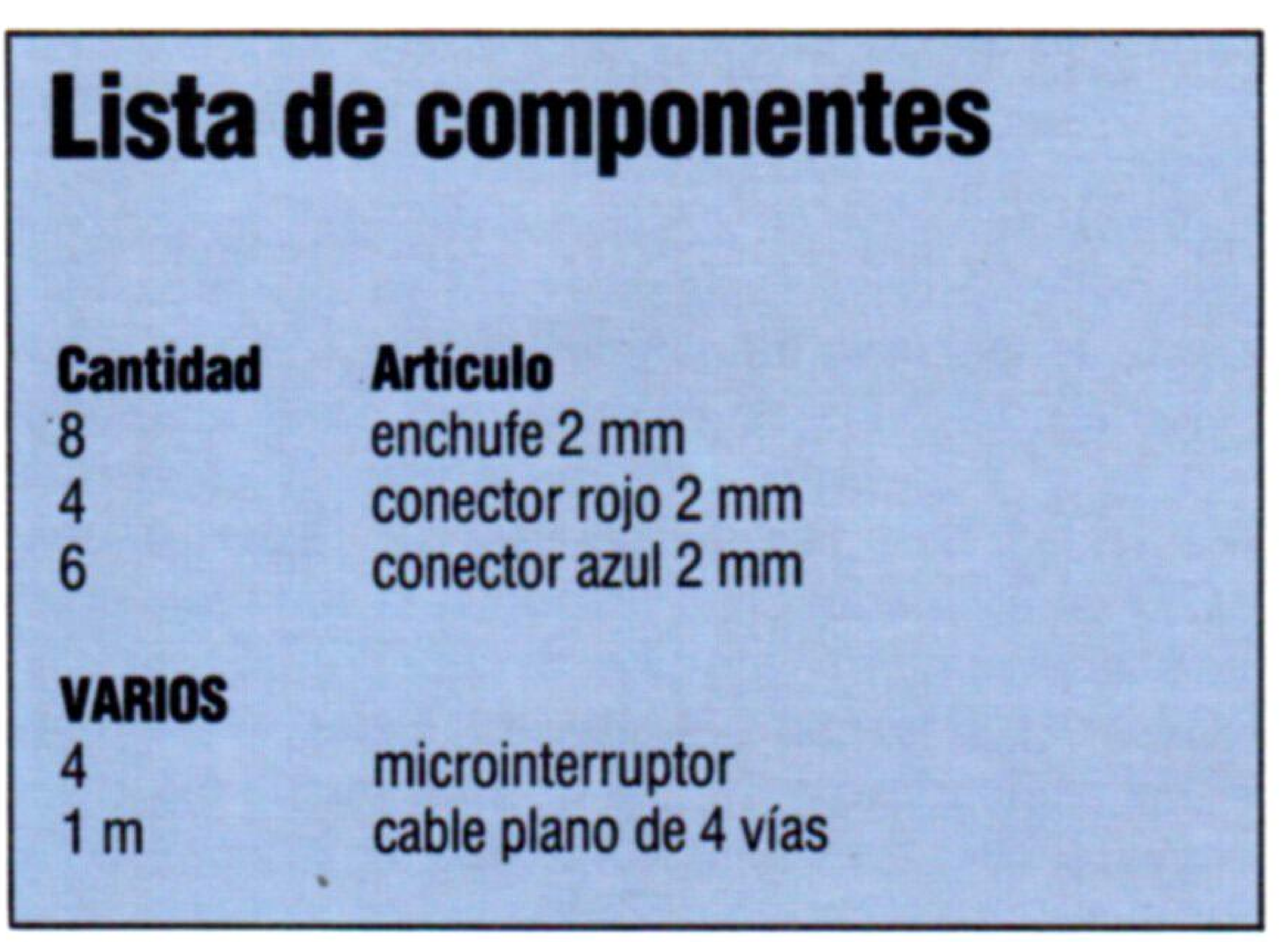
El circuito

El circuito que conecta los sensores de microinterruptor con el sistema del robot es muy sencillo. Las líneas de datos de D4 a D7 están conectadas a los 4 conectores montados en la tapa del robot; las líneas de datos de D0 a D3 se utilizan para el control del motor. Uno de los lados de cada microinterruptor está conectado a un grupo similar de 4 conectores; el otro lado va conectado a una tierra común. De requerirse los 4 interruptores, se los puede parchear en las líneas de datos mediante 4 cables de parche. Si los 4 bits superiores del registro de datos de la puerta para el usuario se establecen en entrada, entonces se suelen mantener altos (en 1). El cierre de cualquier microinterruptor parcheado en el sistema conectará a tierra la línea de datos, poniendo bajo (en 0) el bit correspondiente





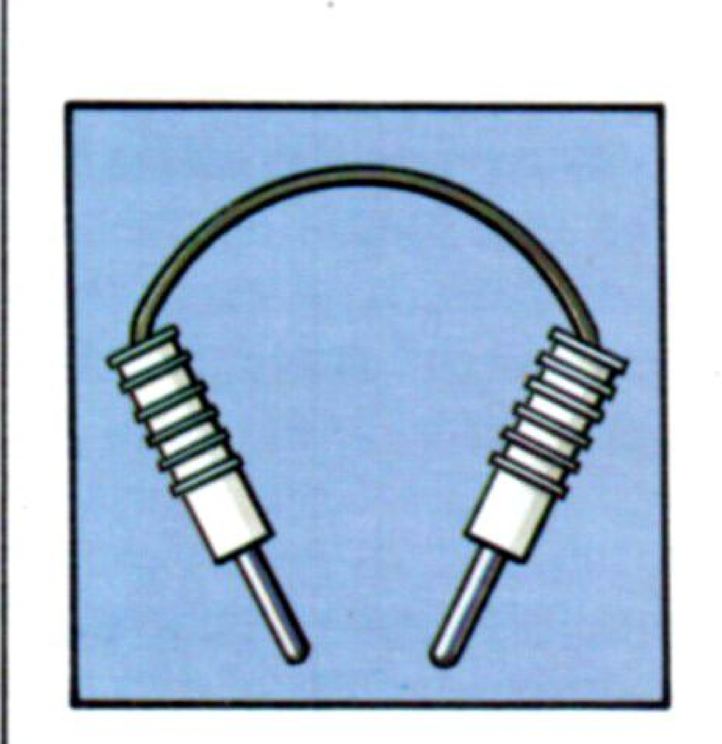


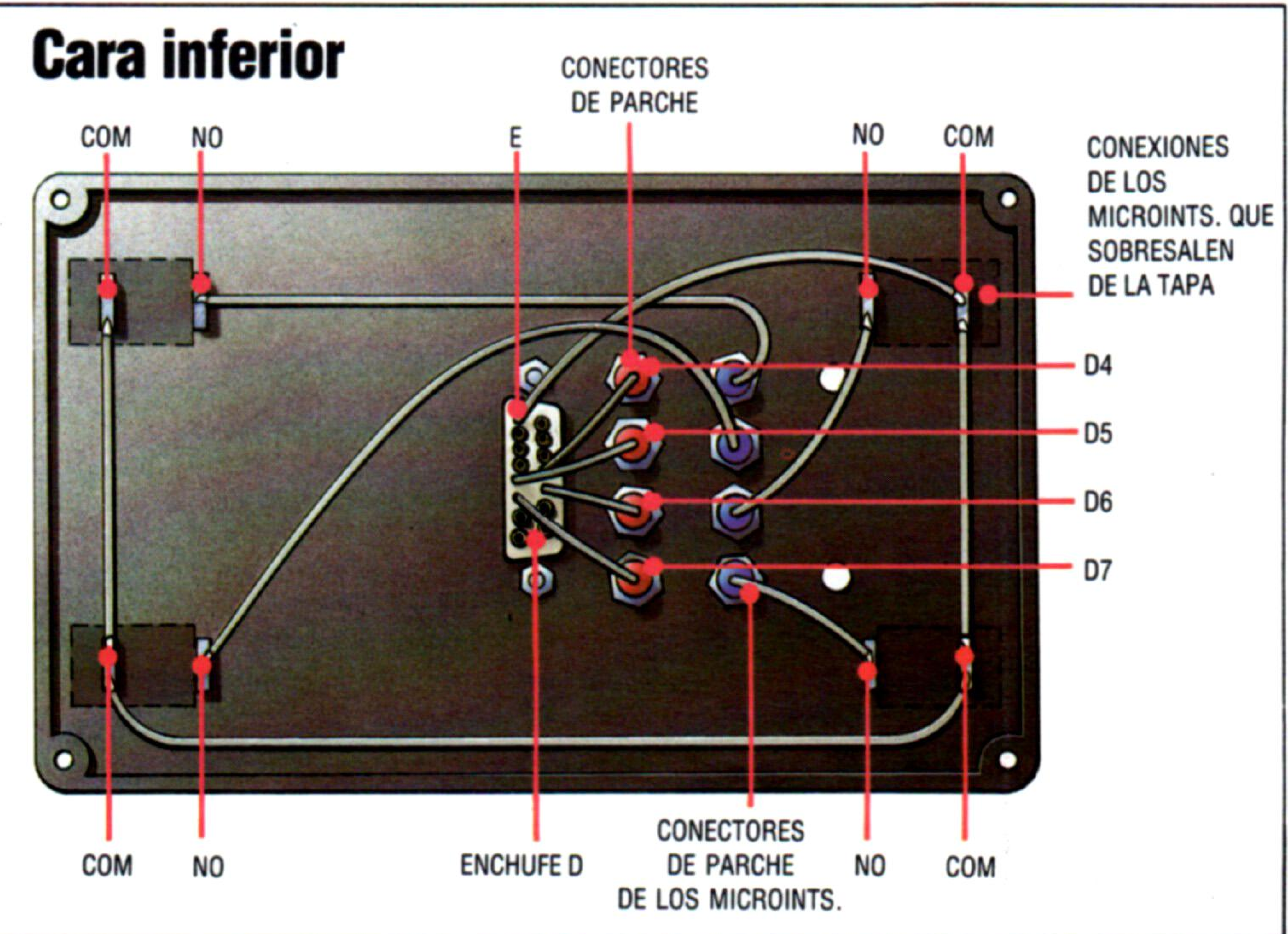


Lista de componentes

Cantidad	**Artículo**
8	enchufe 2 mm
4	conector rojo 2 mm
6	conector azul 2 mm
VARIOS	
4	microinterruptor
1 m	cable plano de 4 vías

Con los microinterruptores ya montados, dé la vuelta a la tapa. Suelde trocitos de cable revestido a cada uno de los 4 conectores rojos y conecte éstos a las patillas del enchufe D. Los 4 conectores COM deben estar unidos entre sí, y la patilla de tierra del enchufe D a un circuito circular. Luego conecte cada conector NO al conector azul adecuado. Construya 4 cables de parche para utilizar con el sistema





Mapa de memoria del BBC

El presente capítulo está dedicado al empleo que el OS del BBC Micro hace de la memoria

El BBC Micro utiliza varias áreas de la memoria, muchas de ellas con varias funciones. Un ejemplo típico es la página &9 de la memoria (o sea, el área de la RAM que va de la posición &900 a la &9FF). Ésta sirve, según las veces, como buffer de salida del RS232, buffer de salida de la cassette, zona de trabajo para la voz y de zona de trabajo para el sonido ampliado. ¡Nadie puede acusar a Acorn de despilfarro de memoria!

Otro ejemplo es el bloque que va desde &0E00 a &1900. Esta área, cuando se emplea la cassette como sistema de archivo, queda libre para ser usada como espacio libre para programas BASIC. Pero si lo que se emplea es el disco, sirve de zona de trabajo del sistema de archivos en disco, reduciendo el total de memoria reservada a los programas BASIC en más de 2,5 K. Lo cual es una dificultad con los Modos del 0 al 2, pues ya queda previamente reducida la memoria.

Un último ejemplo del multiuso de la memoria está en el área entre &8000 y &BFFF. Si se acopla una ROM DFS, un procesador de textos o una ROM de utilidades en la máquina, la zona que ocupará será ésta. Pero también es aquí donde reside la ROM del intérprete de BASIC. Éstas se llaman ROM paginadas y sólo una de ellas puede estar activa a la vez. La ROM requerida es seleccionada por el sistema operativo, quien la "conecta" o "pagina". Normalmente la ROM del BASIC ya está paginada y el intérprete funcionando, lo que nos permite escribir y ejecutar programas en BASIC. Pero cuando se ejecuta la instrucción DFS, el sistema operativo pagina la ROM del DFS, ejecuta la instrucción y vuelve a paginar la ROM del BASIC. Esta última sencillamente se ignora durante la ejecución de la orden DFS. Otras ROM tales como las del TFS (Telesoftware Filing System), también para archivos, o del chip procesador de textos Wordwise, también ocupan el mismo espacio de memoria y son paginadas por el OS según se las requiera. Probablemente debido a que el BBC Micro emplea la misma área de memoria para múltiples fines, queda algo más de memoria para el usuario.

Memoria del usuario

Una vez que el OS ha tomado posesión de su parcela de memoria, el resto es la memoria disponible para el usuario. La variable BASIC llamada PAGE (página) contiene la dirección de inicio del área para programas BASIC, y la variable HIMEM (*high memory*: parte alta) señala el inicio de la memoria para visualización en pantalla; el espacio intermedio es el que queda para los programas BASIC, las variables y los programas en código máquina. Digite esta instrucción para conocer dichas direcciones y la memoria disponible para los programas BASIC:

```
PRINT PAGE,"HIMEM"(HIMEM-PAGE) "BYTES LIBRES"
```

En la tabla siguiente se proporciona una breve descripción de los distintos usos de las diversas áreas de almacenamiento para código máquina.

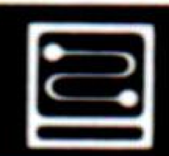
Qué áreas están disponibles para programas en código máquina	Tipo de sist. de archivo	
	Casset.	Disco
&0D00 – &0DFF Página &D: sólo cuando no existen ROM paginadas en la máquina	(✓)	X
&0B00 – &0CFF Pgs. &B y &C: cuando no se emplean en el programa caracteres definidos por el usuario o teclas de función del usuario	(✓)	(✓)
&0A00 – &0AFF Página &A: cuando no se está empleando una interface en serie	(✓)	(✓)
Espacio guardado en el área para programas BASIC por la instrucción DIM		
✓= Sí (✓) = Sí, con reservas X No empleado		

El almacenamiento del código máquina en el área para programas BASIC es la mejor solución para muchas rutinas empleadas junto con programas BASIC.

Una vez introducido en la memoria el código máquina, se necesita por lo general una zona de trabajo para poderlo ejecutar. Muchos programas en código máquina para el 6502 exigen la página cero como zona de trabajo, principalmente porque algunas instrucciones de direccionamiento indexado necesitarán posiciones de la página cero. Acorn ha empleado largamente esta página para el OS, pero algunas posiciones se han reservado para la programación en lenguaje máquina. No es de gran utilidad para el usuario una descripción byte por byte de lo que hace cada una de las posiciones del OS, y no se recomienda el acceso directo a ellas: las posiciones útiles deben accederse por medio de llamadas al OS, lo que significa una fuente de problemas a la hora de ejecutar un programa en código máquina escrito para una versión diferente del OS.

Entrada al OS

Las dos rutas principales que podemos tomar para el empleo de las rutinas contenidas en el OS del BBC son las rutinas OSBYTE y OSWORD:



• OSBYTE nos permite modificar el comportamiento de muchas rutinas del OS, pasando códigos de control y parámetros en los registros A, X e Y del 6502. En BASIC podemos acceder a las rutinas OSBYTE a través de la instrucción *FX. A la instrucción *FX siguen dos o tres números: el primer número es el control o código de función que se pasa al registro A; el segundo es el número que se pasa al registro X, y el tercero el que se pasa al registro Y. El tercer parámetro no es necesario en todas las llamadas de OSBYTE. En código máquina esta rutina es llamada en la dirección &FFF4. Estas dos versiones tienen funciones equivalentes:

BASIC	Assembly
*FX4,1	LDA #4 LDX #1 JSR &FFF4

El valor contenido en el registro A define exactamente lo que hará una llamada determinada a OSBYTE. La llamada ilustrada, por ejemplo, afecta a la acción de las teclas del cursor; el parámetro colocado en el registro X especifica el que las teclas retengan su función editora normal o el que den simplemente un código ASCII.

Desgraciadamente no todas las rutinas OS pueden ser sometidas al OSBYTE. La mayor desventaja está en que el número de parámetros que pueden pasarse a la rutina es limitado. Si se desconoce el contenido del registro A, que es el que indica al OS qué rutina deseamos emplear de entre las de OSBYTE, sólo podemos pasar dos parámetros a los registros X e Y. Para pasar algo más que esto nos serviremos de la segunda rutina, la OSWORD.

• La OSWORD permite cosas tales como generar sonidos, leer y escribir sobre disco, etc. Aquí reside la diferencia entre la OSBYTE y la OSWORD. Aquélla se ocupa de la *manera* en que el OS realiza ciertas tareas, y ésta nos permite la realización concreta de tales tareas. OSWORD obtiene sus parámetros de un *bloque de parámetros* al que apuntan los registros X e Y del 6502 al ceder control a la rutina OSWORD. Este bloque de parámetros se sitúa en la RAM y su tamaño y organización dependen de la llamada OSWORD que se haga. El registro A contiene un código de función que determina cuál de entre las funciones de OSWORD ha de ser ejecutada por el OS. Una vez preparados registros y bloque de parámetros, la llamada a OSWORD se realiza en la dirección &FFF1. Los principales instrumentos de entrada del OS son, sin duda, OSBYTE y OSWORD, y por su singular importancia habremos de examinarlas más adelante con mayor detalle.

Desde BASIC se puede acceder a otras rutinas del OS mediante un asterisco (*) seguido de una instrucción. Este símbolo hace que la instrucción que sigue evite el intérprete de BASIC y sea pasada a una rutina OS de nombre OSCLI (*O*perating *S*ystem *C*ommand *L*ine *I*nterpreter). Esta rutina se encarga de interpretar las órdenes OS digitadas llamando a las adecuadas rutinas del OS. Con frecuencia tales instrucciones se denominan *instrucciones estrella* o *instrucciones* *. El cuadro adjunto al final del capítulo da una relación de tales instrucciones reconocidas por el BBC; las que no son reconocidas, sea porque no se escribieron correctamente o porque

Mapa de memoria del BBC Micro

&FFFF ROM del OS; &FF00 Mapa de E/S; &FC00 ROM del OS; &C000 ROM del BASIC; &7FFF RAM del video; Pila del BASIC; HIMEM; Tope de variables ?2+256*?3; Variables; Tope; Programas BASIC; &1900 Página para sistema; Espacio prog. BASIC o de trabajo para el DFS; &0E00 Página; Zona trabajo en ROM paginada o cód. máq. usuario; &0D00 Caracteres def. usuario; &0C00 Teclas def. usuario; &0B00 Buffers (memorias transitorias); &0900 Espacio trabajo; &0800 Espacio trabajo para BASIC; &0400 Espacio trabajo; &0300 Espacio trabajo; &0200 Pila del 6502; &0100 Página cero; 0

&FEFF Celia; &FE00 Toni; &FD00 Juan; &FC00

ROM paginadas; Tope de la RAM

&FEFF; &FEC2 Chip ADC; &FEC0; &FEA3; 6854; &FEA0; &FE84; 8271 Controlador discos flexibles; &FE80 Vía usuario; &FE60 Vía sistema; &FE40; Selec. ROM pag.; &FE30; &FE21; ULA video; &FE20; 6850; &FE08; &FE01; 6845 CRTC; &FE00

&3FF Buffer teclado; &3DF Variables del sistema cassette; &37F Esp. trabajo VDU para texto/gráf.; &2FF Esp. trabajo sist. archivo y cassette; &2E9 Espacio trabajo del OS y variables escritas por llamadas FX; &235 Vectores; &200

Liz Dixon

no se acompañaron del número adecuado de parámetros, suelen provocar un mensaje de error del tipo Bad Command (orden impropia).

Podemos pasar órdenes a la CLI (*C*ommand *L*ine *I*nterpreter: intérprete de línea de instrucción) por medio de la rutina OS o bien directamente. Los servicios de la OSCLI son dobles: primero, permite que pasemos las instrucciones del citado cuadro a la CLI vía código máquina si así lo deseamos, y, segundo, nos permite pasar variables alfanuméricas en BASIC a la CLI. Los siguientes programas ilustran ambos usos. Obsérvese que las variables numéricas enteras, A%, X% e Y% pasan sus valores directamente a los registros A, X e Y. De ellas, X% e Y% (o, lo que es lo mismo, los registros X e Y) apuntan a la

posición de memoria de la cadena de caracteres que se ha de tratar como una instrucción asterisco, y debe ser interpretada y ejecutada por la rutina OSCLI. Mientras X retiene el byte *lo* (inferior) de la dirección e Y retiene el *hi* o superior.

Versión BASIC

```
10 DIM C 100
20 OSCLI=&FFF7
30 INPUT"ENTRAR
   INSTRUCCION",A$
40 $C=A$
50 X%=C MOD 256
60 Y%=C DIV 256
70 CALL OSCLI
80 GOTO 30
```

Versión BASIC+Assembly

```
 10 DIM C 100
 20 OSCLI=&FFF7
 30 FOR I%=0 TO 2 STEP 2
 40 P%=&C00
 50 [ OPT I%
 60 .code   LDX #C MOD 256
 70         LDY #C DIV 256
 80         JSR OSCLI
 90         RTS
100 ]:NEXT I%
110 INPUT "ENTRAR INSTRUCCION ", A$
120 $C=A$
130 CALL code
140 GOTO 110
```

La ejecución de este programa produce un aviso: el usuario digita una instrucción *, pulsa la tecla Return y la instrucción se ejecuta. La sentencia, más bien extraña, de la línea 10 de ambas versiones, DIM C 100, hace que el ordenador reserve 100 bytes de memoria, en el espacio reservado para las variables BASIC e inicializa la variable C con la dirección del inicio del bloque de memoria. Estos 100 bytes pueden ahora emplearse para almacenar programas en código máquina o, en un caso como éste, los datos de los programas en código máquina. La sentencia $C=A$ coloca los bytes que componen la cadena de la instrucción contenida en A$ dentro de ese bloque de 100 bytes, comenzando por el primer byte reservado por DIM. En ambas versiones los registros X e Y (o sea, las variables X% e Y%) son activados realizándose así la llamada a la OSCLI. La cadena de la instrucción se ejecuta seguidamente.

Este programa es la base de una rutina que se emplea en programas accionados por menú, donde puede resultar útil permitir al usuario obtener un catálogo de discos o cintas, por ejemplo, sin tener que abandonar el programa. La instrucción requerida se coloca sencillamente en las variables alfanuméricas y se pasa a la OSCLI para ser ejecutada. Si digitamos *A$ no funcionará. El OS intentará ejecutar una instrucción llamada *A$ que no existe.

Por medio de esta técnica es también factible pasar variables numéricas a una instrucción * mediante la función STR$ para convertirlas en alfanuméricas. Normalmente la CLI no aceptará ningún nombre de variable que se le pase; produce el mensaje de error Bad Command.

Toda instrucción * que no es reconocida por el OS es pasada a cualquier ROM que esté paginada. A cada una se le pregunta si la reconoce; si es así, la ejecuta. Las instrucciones que no se pasan de esta manera para ser ejecutadas son tratadas como órdenes incorrectas *sólo si* no se emplea un sistema rápido de archivo, como, por ejemplo, la unidad de disco. Si se emplea, el disco será inspeccionado para ver si contiene un archivo con el mismo nombre que el de la instrucción (excluido el *). Si lo contiene, el archivo se carga en la máquina y es tratado como un programa en código máquina. Esto puede ocasionar numerosos problemas en caso de que tal archivo no sea en realidad un programa. El ordenador suele "insistir" en estas situaciones hasta que usted no llega a sacarlo de su atasco. Si tal archivo no existe, entonces visualiza un mensaje de error Bad Command.



Uso de la página cero en el BBC Micro
Las limitadas facilidades de indexación del 6502 se complementan con su uso de la página cero (&0000 hasta &00FF) como una matriz de registros índices auxiliares. Esto conduce naturalmente a una densa utilización de estas posiciones por el OS, pero existen unos pocos bytes que no se emplean y se dejan para uso del programador

Las instrucciones*

Instrucción	Descripción y comentarios
*HELP	Da el número de la versión del BASIC. Puede también servir para obtener información sobre otras ROM paginadas que se han incorporado: así, *HELP DFS
*BASIC	Da entrada al lenguaje BASIC. Una variante de esta instrucción es, por ejemplo, *WORD, que da entrada a la ROM paginada View, en este caso
*CODE *LINE	Sólo en versión 1.2 del OS. Permite añadir nuevas órdenes
*KEY	Para programar teclas función rojas
*MOTOR n	Para controlar el relé del motor de la cassette: n=0 desconecta el relé y n=1 lo conecta
*ROM, *TAPE, *DISK, *NET	Para inicializar el adecuado sist. de archivo: *TAPE dará entrada al sist. de archivo en cass. de 1 200 baudios, y *DISK al de archivo en disco
*FX	Permite que el programador controle los valores de varias variables del OS y, gracias a eso, el comportamiento del OS
*RUN, *OPT *LOAD, *. *CAT, *SAVE	Instrucciones del sistema de archivo que examinaremos más tarde en otro capítulo
*SPOOL *EXEC	*SPOOL envía la salida de pantalla a la pantalla y a un archivo. *EXEC lee datos de un archivo como si se leyeran del teclado
*TV x,y	Afecta a la posición vertical de la pantalla y la interface de pantalla: x=0 no produce cambios en la posición vertical; x=1 mueve la pantalla hacia abajo una línea; x=255 la mueve hacia arriba una línea; y=0 activa el "interlace"; y=1 lo desactiva. Sus efectos se hacen operativos en el sig. cambio de modo y no cesan hasta que no se oprima CTRL-BREAK u otra instr. *TV

Una más detallada explicación de estas instrucciones las obtendrá el lector en la misma guía del usuario del BBC Micro

mi computer

Editorial Delta, S.A.



10068
9 788485 822836